University of Alberta

System Architecture for Internet-based Teleoperation System

by

Teresa Tak-Sum Ho

A thesis submitted to Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta

Fall 1999

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-47040-7

Canada

University of Alberta

Library Release Form

Name of Author: Teresa Tak-Sum Ho
Title of Thesis: System Architecture for Internet-based Teleoperation System

Degree: Master of Science
Year this Degree Granted: 1999

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Teresa Ho
.....

Teresa Tak-Sum Ho
57 Langholm Drive
St. Albert, Alberta
Canada T8N 4N9

Date: Sept. 27, 1999

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **System Architecture for Internet-based Teleoperation System** submitted by Teresa Tak-Sum Ho in partial fulfillment of the requirements for the degree of **Master of Science**.

.....
Dr. Hong Zhang

.....
Dr. Max Meng

.....
Dr. Benjamin Watson

Date: Sept. 20. 1999

Abstract

The Internet has become part of everyone's daily life. It connects people worldwide and provides a means of affordable communication. In order to take advantage of this ever-popular medium, researchers have started to use it as a medium for remote control of a robot. Such remote operation can replace the need of human presence at hazardous or unreachable location. Current research has concentrated on the aspect of functionality and researchers have neglected to take advantage of the ever-changing Internet technologies to improve the user interface and the backbone of these systems. This thesis takes the first steps in creating an intuitive 3D graphical interface and a superior system architecture for Internet-based Teleoperation Systems. To demonstrate the feasibility of the system architecture, we use a three-fingered hand that can be remotely controlled through the Internet to perform manipulation tasks.

Contents

CHAPTER 1 INTRODUCTION **1**

1.1 INTRODUCTION 1
1.2 GOAL 2
1.3 MOTIVATION 3
1.4 OUTLINE OF THESIS 3

CHAPTER 2 TELEOPERATION AND IBTS **4**

2.1 INTRODUCTION 4
2.2 INTERNET-BASED TELEOPERATION 6
2.2.1 Definition *6*
2.2.2 History *7*
2.3 DESIRABLE CHARACTERISTICS OF AN IBTS 11

CHAPTER 3 ARCHITECTURE OF INTERNET-BASED TELEOPERATION SYSTEMS **12**

3.1 INTRODUCTION 12
3.2 CONTROL INTERFACE FOR THE HUMAN OPERATOR 13
3.3 INTERNET ACCESS METHOD 15
3.4 REMOTE HOST MACHINE 16
3.5 ROBOT AND ITS CONTROLLER 17
3.6 SENSORY FEEDBACK 18

CHAPTER 4 IMPLEMENTATION OF THE THREE-FINGERED HAND SYSTEM **19**

4.1 INTRODUCTION 19
4.2 JAVA 19
4.3 VRML 20
4.4 ARCHITECTURE OF THE THREE-FINGERED HAND SYSTEM 22
4.4.1 Control Interface for Human Operator *24*
4.4.2 Java Servlet-Based Communication *33*
4.4.3 Remote Host Machine *34*
4.4.4 Robot and Its Controller *35*
4.4.5 Sensory Feedback *38*
4.5 OPTIMAL VIEW-ANGLE 40
4.5.1 Defining the Optimal View-angle of the Manipulated Object *40*
4.5.2 Rotation in 3D Space *43*
4.5.3 Equation for a Plane Surface *45*
4.5.4 Back Face Elimination *47*
4.5.5 Z-buffering *50*
4.5.7 Finding the Optimal View-angle *51*
4.6 ROBOT KINEMATICS 55

CHAPTER 5 EVALUATION CRITERIA **59**

5.1 INTRODUCTION 59
5.2 VIDEO QUALITY AND SPEED 59
5.3 ACTUAL EXECUTION TIME OF THE ROBOT 61
5.4 NUMBER OF EXECUTION ERRORS 62
5.5 RELIABILITY – 24 HOURS A DAY, 7 DAYS A WEEK 63
5.6 SUBJECTIVE EVALUATION CRITERIA 64

CHAPTER 6 CONCLUSION **65**

6.1 SUMMARY 65
6.2 FUTURE RESEARCH 66

BIBLIOGRAPHY 69

List of Figures

Figure 1. An overall picture of a teleoperation system. 5
Figure 2. Relationships between basic components of an IBTS. 6
Figure 3. A snapshot of the control interface of the Mercury Project. 7
Figure 4. System layout of a typical Internet-based Teleoperation System. 13
Figure 5. Example of perspective and parallel projections respectively. 22
Figure 6. System layout of the three-fingered hand system. 23
Figure 7. Snapshot of the login screen. 25
Figure 8. A snapshot of the control interface of the three-fingered hand system. 26
Figure 9. Labels of all the objects in the VRML model. 27
Figure 10. Limit of finger 2, joint a. 27
Figure 11. Limit of finger 2, joint b. 28
Figure 12. Shows the two shift nobs of the view-angle controller and the x, y and z axis in VRML. 28
Figure 13. Snapshot of the Java-based controller. 30
Figure 14. The initial positions of all fingerjoints. 32
Figure 15. Flow diagram of how Java Servlets are served to a user through the web browser. 34
Figure 16. Picture of the three-fingered hand. 35
Figure 17. Picture of servomotors of different sizes. 36
Figure 18. Picture of the SSC board. 36
Figure 19. Diagram of the two SSCs, the hand and the remote host machine. 37
Figure 20. The overall picture of all imaging components. 39
Figure 21. Three view-angles of a prism. 41
Figure 22. Different view-angles show different portions of the 3 visible surfaces. 42
Figure 23. Rotation of a point. 43
Figure 24. Surface A, B and C are not removed by back face elimination. 47
Figure 25. Example of a non-back face. 49
Figure 26. Example of a back face. 49
Figure 27. A z-buffer of size 100x100 and its content of the colored faces of each of the 3 shapes. 51
Figure 28. Optimal view-angle for the rectangular-prism. 53
Figure 29. 10° from the optimal view-angle. 53
Figure 30. Not-so-optimal view-angles. 54
Figure 31. Contact points of the three fingers and the rectangular-prism. 55
Figure 32. Labels of the joint angles and the lengths of the finger links. 57

Chapter 1

Introduction

1.1 Introduction

In today's world, the Internet plays an important role in people's lives. It provides a convenient channel for receiving information, electronic communication and conducting business. Robotics researchers are now using the Internet as a means to provide feedback for teleoperation [Sheridan]. Internet-based teleoperation will inevitably lead to many useful applications in various sectors of society. For example, it is impractical to transport skilled surgeons from one side of the world to another to perform life-saving operations. However, with the help of the Internet, surgeons may one day remotely operate on patients despite the distance between them. To create such highly reliable Internet-based teleoperation systems (IBTS), the foundations must be strong. Further, the different technical issues must be examined in detail. In the short history of Internet-based teleoperation, a number of experiments have been conducted [Atkinson, Backes, Carnegie, DePasquale, Digital, Fisher, Goldberg, K., Goldberg, S., Saucy, Wolf]. Since the Mercury Project [Goldberg, K. 1994], which allows users to excavate objects in a sandbox, the users of Internet-based teleoperation are often general Internet users who seek recreational amusement [Carter]. These users are given tasks like demolishing and

stacking wooden blocks as in the Australian Telerobot [Taylor, 1995]. Later, the Tele-Garden [Goldberg, K. 1997] was also built to let Internet users tend a garden that contains live plants. More recently, an interactive model railroad [Wolf] is built to let users from all over the world control two train sets. For the artistic Internet users, museum exhibits are set up for viewing via a remotely controlled camera [Goldberg, S.]. As well, a remote painting system has been built by the PumaPaint Project [DePasquale]. The Jason Project [Fisher] allows users to seek adventures by solving on-line mysteries through the teleoperation system. Spying can also be performed on a remote site via the Eyebot [Digital] - a camera that can be controlled to observe a remote environment. The above examples show that on-line teleoperation systems are often constructed for recreational purposes. Users are not robotics experts but rather general Internet users with "average education level [that] has been declining to be more representative of the general population" as observed in the April 1998 WWW Survey Results [Kehoe].

1.2 Goal

Through recreational-oriented experiments, various functional components of an IBTS have been identified and different implementation techniques have been employed. As a result, an architectural standard has emerged which defines the functional components and their interfaces. The goal of this research is to be the first to present a solid system architecture for IBTS. We will also make improvements to components of this architecture. First, we want to apply advance Internet technology to establish a secure yet flexible system. Second, we will demonstrate the design and implementation of a new input format. To enhance usability, this robotic research also takes advantage of graphic

theories to provide user with an automated view controller. All these improvements are significant to the future advance of research in IBTS.

1.3 Motivation

Through Internet-based teleoperation systems, human lives can be saved because it eliminates the need of human to be present in hazardous environments like minefields and nuclear factory. Also, resources can be shared between users around the world. For example, if we set up a three-fingered hand at the University of Alberta, researchers in Australia can time-share and use the hand for their own research. This way, many users can share expensive and hard-to-find robots. The use of the Internet has also become very popular worldwide and thus, it is an inexpensive two-way communication link that spans across oceans and around the world.

1.4 Outline of Thesis

This thesis first outlines the popularity of the Internet and then points out that many researchers have found it to be a suitable medium for teleoperation. It is followed by definitions and examples of Internet-based teleoperation system (IBTS). Chapter 3 shows details of the architecture of an IBTS and each component of the architecture is explained. Chapter 4 contains specific details of the multi-fingered hand IBTS. It also shows how to find an optimal view-angle direction for the multi-fingered system. Evaluation criteria of IBTSs are then presented in Chapter 5. Finally, this thesis will end with a conclusion and a set of proposed future work.

Chapter 2

Teleoperation and IBTS

2.1 Introduction

To understand the meaning of teleoperation, first we will examine the definition of robotics. Robotics is the science and art of designing and using robots [Sheridan]. Robot is further defined as "a reprogrammable multi-functional manipulator designed to move materials, parts, tools or specialized devices through variable programmed motions for the performance of a variety of tasks" [Sheridan]. Furthermore, any electromechanical systems, such as a toy train, may be classified as a robot because it manipulates itself in an environment.

Teleoperation is the direct and continuous human control of a teleoperator. A teleoperator can be any machine that extends a person's sensing and / or manipulating capability to a location remote from that person. Edwin G. Johnsen and William R. Corliss first defined the term teleoperator in 1971 [Johnsen]. The prefix "tele" in teleoperator describes the ability of this class of machine systems to project man's innate dexterity not only across distance but through physical barriers as well. In situations where it is impossible to be present at the remote location, a teleoperator can be used

instead. These situations may be caused by the hostile environment at a minefield, or at the bottom of the ocean or simply at location that is impossible to travel. A teleoperator can replace human in any hazardous environments. Histories have shown that teleoperators have gone underwater to survey environment in the early 1970s [SSC], have translated through air and provided short range aerial surveillance in early 1980s [SNWSC] and have been in minefield breaching operations in 1997 [Richardson].

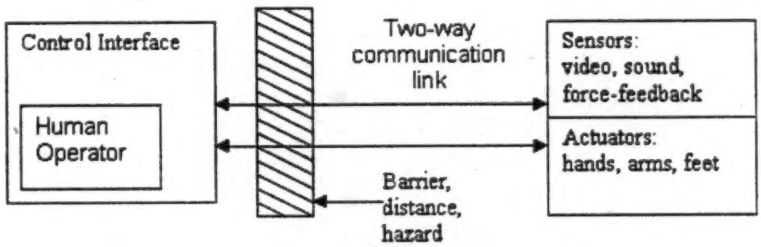

Figure 1 . An overall picture of a teleoperation system.

The above diagram shows the overall picture of a teleoperation system. It illustrates how a human operator can use a control interface to control an actuator that is behind a barrier, away in distance or in a hazardous environment. In between the human operator and the actuator stands a two-way communication link. Through this link, control instructions are sent to the actuator and sensor feedback is also sent back to the control interface for display to the human operator. For distances that are within reach, this link can be simple wire cables. To control flying actuators or those actuators that are not reachable with direct cables, we can use radio signals. The disadvantage of using radio signals is that the controller and the actuator robot has to be within range. This limitation can be overcome by using the Internet as the two-way communication link. In such cases, a human operator can easily control an actuator that is on the other side of the

world. Such a remote manipulation system is called Internet-based teleoperation system (IBTS).

2.2 Internet-based Teleoperation

2.2.1 Definition

To remotely control a robot through the Internet, the human operator needs to be connected through its Internet Service Provider (ISP). Then the operator can use a web browser to access the control interface that is contained in a web page. This web page is simply a text file that is located on a web server. The following diagram shows the route of how the web-based controller interface is accessed by a human operator through the Internet.

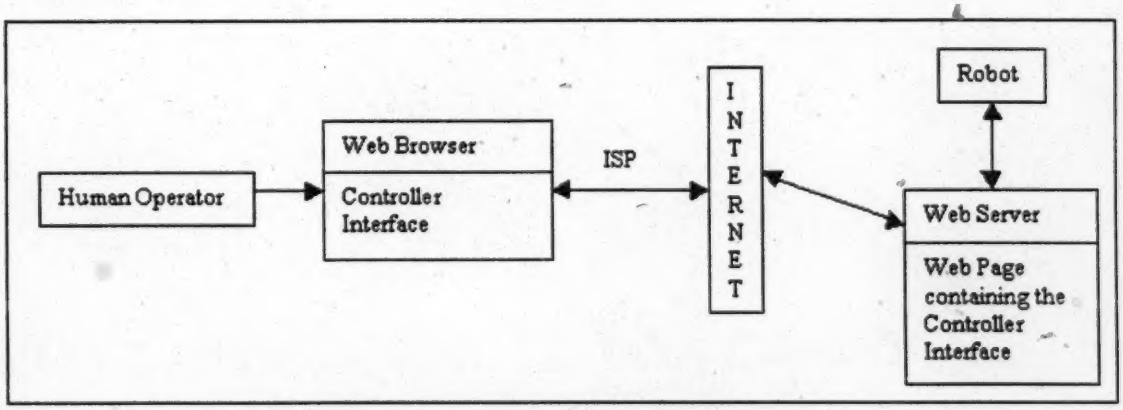

Figure 2. Relationships between basic components of an IBTS.

A web browser is a computer application that can be used to view web pages and it is executed on the human operator's local computer. On the other hand, a web server is located at the remote site where the robot is. It contains the controller interface and sends the interface to the remote human operator upon request. Finally, the human operator is

hooked up to the Internet through his or her ISP. In this set-up, the Internet allows users to remotely control a robot that is not within reach.

2.2.2 History

The first IBTS was created by the Mercury Project [Goldberg, K. 1994]. It consisted of an industrial robot arm that carries an air pump. By moving the arm to the various locations in a sandbox and then exploring it with the air pump, different objects can be revealed in the sandbox. Attached to the arm is a video camera that provides visual feedback to the teleoperator. This system was built as a game; thus, in the first 5 months, it attracted over 50,000 unique users around the world. The following diagram illustrates the user interface for the system:

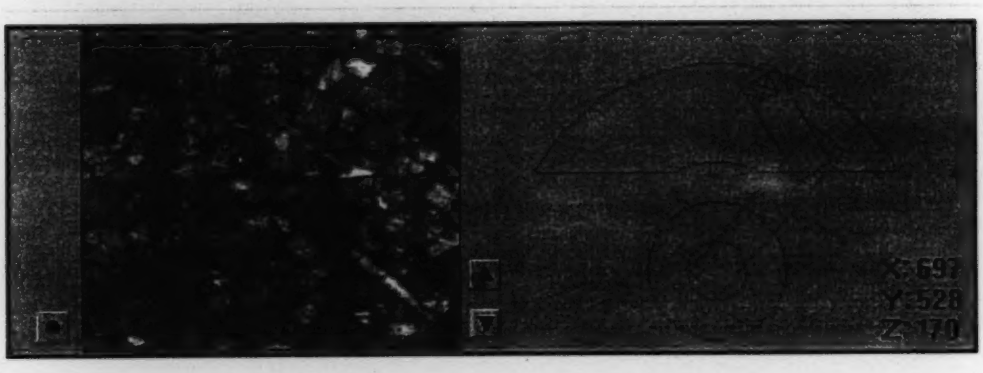

Figure 3. A snapshot of the control interface of the Mercury Project.

The control is accessed via a web browser. As seen in Figure 3, it is made up of a live video feedback display and a two-dimensional clickable controller. The clickable controller shows a 2D drawing of the robotic arm and its placement relative to the sandbox. The human operator can click on any location of the clickable image and the

control instructions will be sent to the robotic arm to move to the corresponding location of the sandbox. Then using predefined buttons that are located in the controller, a user can issue instructions to lower or raise the robotic arm and blow air into the sandbox and thus, discover the different objects that are in the sandbox. This system showed that an Internet-based teleoperation system is possible and the system was enjoyed by more than 50,000 users worldwide.

The second IBTS was from the Robotics & Automation Lab in Western Australia. It was brought on-line only a few weeks after the first system [Taylor]. This system allows an Internet user to control a gripper that manipulates wooden blocks. The control interface is similar to that of the above system. It is in the form of a web page. It consisted of a live video display that shows the status of the remote environment. The controllers, themselves, are a set of buttons and text inputs that require the user to select and specify the next movement of the gripper. This controller interface requires a user to think in low-level details; for example, it requests the x, y, z coordinates of the next move. It is not as easy to use as the previous system.

The Tele-Garden [Goldberg, K. 1997] was also built to let Internet users participate in tending a living garden using a remote industrial robot to perform simple requests such as watering, planting and viewing the garden. The human operator accesses the web page that contains the controller and by clicking on the 2D representation of the garden, the user can explore the garden. A live video display of the garden is also presented as part of the controller. Predefined buttons are likewise provided to perform a set of tasks like watering and planting. The control interface was very intuitive to use and users frequently return to monitor the growth of their planted

seeds. Thus, the traffic of this IBTS was heavy and steady throughout its operational years.

In 1996, an interactive model railroad [Wolf] was built to let users from all over the world control two train sets. A live video displays the current location of the trains and shows the different train stations that are available. The user simply clicks on the web page that contains the controller to specify the train that is to be moved and the destination of the train. Since 1996, there has been an average of 100 users per day.

All of the above systems were for entertainment. In order to attract the artistic users of the Internet, museum exhibits are set up for viewing via a remotely controlled camera [Goldberg, S.] in 1996. The interface of the controller was very easy to handle. To move the camera's position, the user only needs to drag and drop on a 2D graphical representation of the camera. A live video shows continuous images of a museum exhibit and as soon as the camera is repositioned, the view-angle of the images is changed respectively. The motive behind this IBTS is debatable because the museum exhibit is static and does not change over time. There is no need for real-time inspection of the exhibit. The images or live video could be pre-captured and play back accordingly as the user reposition the camera to a different view-angle.

Another interesting artistic IBTS is a remote painting system called the PumaPaint [DePasquale]. In 1998, PumaPaint was brought on-line. The system consisted of a robotic arm that can pick up a paintbrush and then dip into different colors of paints. The user can then instruct the painting arm to draw on a canvas. However, it is worth mentioning that the interface for this system is different from the others. It does not allow the human operator to control the movement of the robotic arm directly. It

simply asks the user to draw using a mouse on a virtual canvas that is being displayed on the web page. Then the system issues corresponding instructions to request the robotic arm to reposition and draw on the real canvas. A set of predefined buttons and scrollers are also used on the controller web page to allow user to select different colors of paint and to use different levels of force when drawing on the canvas. By selecting a heavy force level, a user can "squash" a lot of paint on the canvas and often, it oversoaks the canvas and makes a mess at the remote location where the painting arm is.

Internet users can also seek adventures by solving on-line mysteries through the teleoperation system built in the Jason Project [Fisher] or spy on a remote site using the Eyebot [Digital]. These two IBTSs both involve the manipulation of a camera that can be controlled to observe a remote environment. Also, they show a very simplistic interface that only allows the user to move the camera in a predefined pattern. For example, the camera can only be moved up, down, left and right. The mobility of this camera is limited and these research projects show only primitive control of a remote camera.

From the above teleoperation systems, it is shown that on-line teleoperation systems are often constructed for recreational purposes and thus, it attracts a wide range of users around the world. This is a beneficial factor because the more users a system attracts, the more feedback is received and eventually a larger number of improvements to the system can be made. It is also interesting to note that the NASA Space Telerobotics Program recognizes that research in IBTS helped advance Space robotics technology; thus, it promotes and keeps track of an up-to-date listing of all these systems. This list is at: http://ranier.oact.hq.nasa.gov/telerobotics_page/realrobots.html.

Furthermore, NASA has gathered a selected number of robotics resources on the Internet [NASA].

2.3 Desirable Characteristics of an IBTS

As an Internet-based teleoperation system, it is desirable to be accurate and reliable without sacrificing the cognitive capability and adaptability of the human operator. The control of the system should be fast and unconstrained by the limited pace of the continuous human sensorimotor capability. It should be made easier by letting the operator give instructions in terms of objects to be moved and goals to be met, rather than low-level instructions to be used and control signals to be sent. It should likewise eliminate the demand for the human operator's continuous attention, thus reducing the operator's workload. Furthermore, automated control is needed where there are time delays in communication between the human operator and the control interface. Finally, the system cannot crash if the operator fails to properly control the remote robot. Overall, an ideal teleoperation system should save lives and reduce cost by eliminating the need for the operator to be present in hazardous environments [Sheridan].

Chapter 3

Architecture of Internet-based Teleoperation Systems

3.1 Introduction

Through examining the different teleoperation systems throughout the history of IBTS, we discovered that these systems are all based on a similar system architecture. Figure 4 shows the overall system architecture of a typical IBTS and the relationships between the components. Each component's role will be defined below and examples will follow.

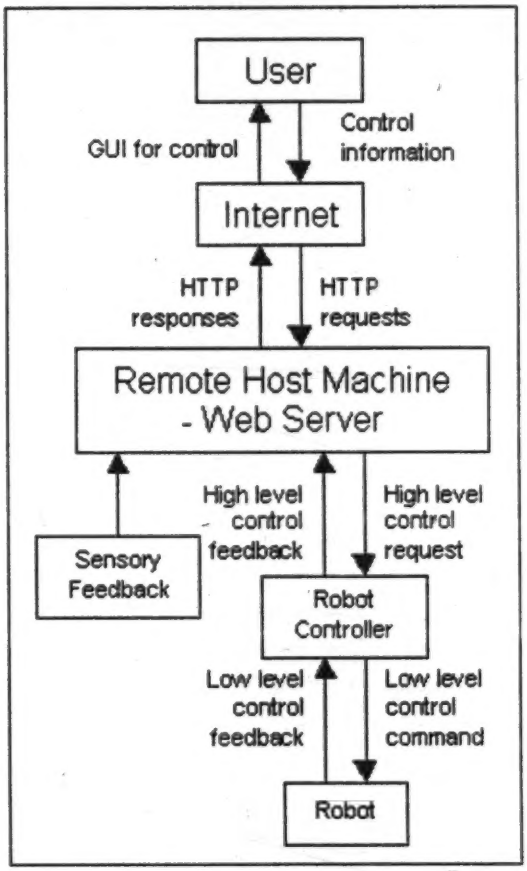

Figure 4. System layout of a typical Internet-based Teleoperation System.

3.2 Control Interface for the Human Operator

Users of any IBTS are required to interact with the remote robot through a control interface. Because these teleoperation systems are Internet-based, the control interface may be served through a web browser or it may be a stand-alone Internet application. A web browser is software that communicates via the HyperText Transport Protocol (HTTP) in order to bring text and graphical information from a web server to the user. An interface that is based on a web browser is created using Hyper Text Markup Language (HTML) and usually combined with server-side Common Gateway Interface (CGI) programs that are written in Perl or C. A stand-alone interface can be built using

any programming tools but it is usually platform-dependent and has to make use of networking protocols to communicate with the remote host to bring information to the user.

The control interface itself, regardless if it is displayed through a web browser or implemented as a separate application, can be in a number of formats. It can be a set of predefined buttons that users can press to trigger predefined actions. The interface can also be a form that requests a user to enter alpha-numeric representation of the next command that controls the robot. Another type of interface makes use of an image that is mapped to a set of predefined actions; depending on which part of the image the user clicks on, a predefined action is triggered. A more intuitive alternative is when the interface is a graphical animation of the real robot and moving the animated robot controls the robot itself.

The railroad system [Wolf] at University of Ulm uses a set of buttons that allows users to select which train to control and which station to park the trains. The Eyebot Project [Digital] also allows users to control a camera through a set of buttons to move the camera up, down, left, right, forward and reverse. A different way to control a robot is shown in the block building system [Taylor, 1995] where users have to enter numbers in the different fields of a hypertext markup language (HTML) form to specify the next x, y and z position of the telerobot. On the other hand, in the Tele-Garden project [Goldberg, K. 1997], a two-dimensional graphical representation of the environment is used to allow a user to click on parts of the graphical image to specify the next location of the telerobot. A 2D draggable graphical interface is also used in the museum exhibits teleoperation system [Goldberg, S.]. User can drag and manipulate a graphical

representation of the camera to position the camera in various view-angles. Currently, a number of research projects use graphical models of the robots to allow the users to control the robot off-line and practice the control techniques. Examples can be found in the RobotToy research [Atkinson], the WITS project [Backes] and the KhepOnTheWeb project [Saucy]. In Talyor's play-with-wooden-blocks teleoperation system [Taylor, K.], 2D wireframe models of the robot are also used to control the actual robot. However, it is found that given a choice between the 2D wireframe model and text-based forms that require users to specify numeric values of the next location, users only select to use the 2D models 4% of the time. This result is disappointing, but it may be due to the fact that 2D models cannot properly represent the 3D world.

3.3 Internet Access Method

The Internet is used as a medium for the remote users to connect to the teleoperation site. Through the Internet, information about the remote environment is sent to the users and commands from the users are sent to the remote host. Applications can either connect directly to a remote host or connect to a web server that services each incoming network request. If a dedicated network connection is to be created, extra security measures have to be taken to verify the data being transferred. Additional set-up has to be established to control the data format for transfer. Also, different levels of connections have to be made to allow for administrative and common users' control of the slave robot. With a network connection, it also makes the remote host machine prone to security attacks. Malicious applications can connect and pretend to perform legal control requests when in reality, it tries to access sensitive information on the host machine, or tries to execute commands to manipulate and destroy the host and its data. In a recent research, DePasquale shows an

example of an IBTS that uses direct network connection to transfer information from the user to the remote host and vice versa. Data packets have to be "checked extensively for validity" [DePasquale]. This validity check creates an overhead in the design and implementation of an ITBS; it causes further delays in the overall system execution. On the other hand, if users only connect to the web server, all interactions are bounded to the security check at the web server itself; no extra set-up is required to verify incoming requests.

3.4 Remote Host Machine

A remote host machine is needed to be present where the slave robot resides. This machine serves as the control center of the IBTS. It also contains the web server. A web server is an advance application that runs on a server and provides connection to remote computers to deliver web page content via the Internet using HTTP protocol. It is used in every IBTS and it should be on-line on a constant basis. Even if dedicated network connections are to be used as the access method, a web server is required to distribute the applications to remote users. The remote host machine is connected to the Internet and an Internet address is assigned to the remote host to accept incoming HTTP requests that are to be serviced by the web server or to accept incoming network connections. Given an HTTP request, a web server responds by retrieving text and graphical information, and sending it back through the Internet to the user.

The type of web server used in previous research varies from Microsoft Personal Web Server [Saucy] to O'Reily's WebSite Server [Fisher] to NCSA HTTP Demon [Goldberg, K. 1994]. The choice of a web server depends on the system platform of the remote host machine and it is subject to the researcher's opinion about the reliability,

security and power of the server. For example, in the Web Interface for Telescience (WITS) Project [Backes, 1997], in order to have a web server of maximum strength, a custom server is built to handle up to 1000 simultaneous requests at the same time. A web server's job is to service incoming HTTP requests. As long as it does its job in an acceptable manner, the choice of the web server has usually been made on the basis of convenience.

Besides hosting the web server and the Internet connections, the remote host also connects to the robot and sensory feedback controllers. By connecting to these controllers, the remote host can retrieve and send information to the controllers as users request them through the web server or the network connection. From the Mercury Project [Goldberg, K. 1994] to the interactive model railroad [Wolf] to the PumaPaint Project [DePasquale], all systems contain a remote host that serves incoming Internet requests from users.

3.5 Robot and Its Controller

Ultimately, the purpose of an IBTS is to allow users to remotely control a slave robot through the Internet; thus, a task-specific robot is used and a controller is created to interface the hardware of the robot with the remote host. The robot can perform a set of tasks. The controller accepts high level control commands from the host machine and translates into low level instructions; it then sends the instructions to the robot. After the robot performs a task, feedback is sent to its controller to report the current status of the robot. This cycle repeats itself as each instruction is sent to the robot.

Depending on the task that is to be performed, different kinds of robots can be used: mobile robot, robotic gripper (two-fingered hand), robotic arm, etc. Each robot

comes with its own controller that serves as the interface between the remote host and the robot. In the KhepOnTheWeb [Saucy] project, a mobile robot is used to allow the users to direct the robot to travel in a maze. A gripper is used to move game pieces in [Carnegie]. An arm with a camera attached to it was used in the Eyebot Project [Digital] to observe a remote environment. The robots can also be any electromechanical systems such as a toy train [Wolf].

3.6 Sensory Feedback

To reflect the current status of the remote environment, different forms of feedback can be provided for the users. The feedback can provide visual, audio, or tactile information of the remote environment. To provide a view of the remote environment, video feedback is often selected and the number of cameras used varies. For example, in the Eyebot Project [Digital], only one camera is used because the purpose of the project is to allow a user to remotely control a camera to survey the remote environment. Often, it is advisable to use multiple points of view. One camera can be set up to provide the global view of the environment like the painting canvas in [DePasquale] and the maze that the mobile robot is travelling through in [Saucy]. Another camera can be used to provide the robot's point of view to the users. In these situations, a camera is mounted on the robot itself like the painting arm in [DePasquale] and the mobile robot in [Saucy].

In summary, we have shown the system architecture of an IBTS. Each component is explained and examples are given. This architecture is the ground work for all IBTS and it is important to test its feasibility. The next chapter shows our implementation of this architecture and it will demonstrate that this architecture is applicable.

Chapter 4

Implementation of the Three-fingered Hand System

4.1 Introduction

In order to understand the detailed implementation of the Internet teleoperated three-fingered hand system, we will first study in this chapter the Internet technologies that are behind it: Java and VRML. The architecture of our IBTS will then follow and each system component will be examined closely. Then, the details of how to find an optimal viewing direction will be discussed. Finally, at the end of this chapter, we will examine the applicable theories of kinematics which are used to calculate the hand's movements.

4.2 Java

Java [Sun, 1998] is a programming language that is platform-independent and object-oriented. Platform independence gives a program the ability to be executed on more than one computer platform, from PC to Mac to Sun Workstations, without the need to rewrite the program's source code. This is also the main benefit of using Java as the programming vehicle for Internet-based applications. The platform independent aspect of the system enables the end-users to be diverse. The users around the world can be on any system and still be able to use the same control interface for the robot. Because the

programs at the remote host machine can also be written in Java, it means that the remote host can be on any machine platform as well. This gives the remote host an ability to be portable.

Java is object-oriented such that a Java-based application can be easily decomposed into a number of objects and these objects can be modified individually without affecting the others. This allows a single system to control different robots simply by rebuilding individual components that are specific to a robot. For example, a system can be switched from controlling an arm to a walking leg by rebuilding components that are specific to the arm and substituting with components that control and interface with the legs of the walking robot. The modifications do not require the rest of the system to be modified accordingly. The changes are modular to each component. The platform-independence and object-orient nature of Java are two features that will benefit any Internet-based application. Our use of Java as the premier programming vehicle is also the first of its kind.

4.3 VRML

Virtual Reality Modeling Language (VRML) is used to construct the graphical model because Java's 3D Application Program Interface (API) was still in beta testing at the time of implementation. However, the specification of the new 3D API is documented and it shows that the API is constructed based on features of VRML [Sun]. As the Java language matures, the model can be implemented purely in Java.

The use of VRML as a temporary 3D modeling solution has introduced other problems because VRML does not have any networking capabilities. In order to send control requests from the VRML model to the web server, Java has to be used as a

medium to send HTTP requests to the web server which issues commands to control the hand [Marrin, 1996, 1997, Pesce]. A link between Java and VRML is created using External Authoring Interface (EAI). This link allows a Java applet to have a two-way communication with a VRML world that is on the same web page. This extra link has caused extra implementation time and execution time of the overall system is hindered also.

VRML is chosen to create a 3D controller mainly because of its capability to convert 2D input to 3D. Normally, the human operator only has access to the control interface through the use of a mouse. Such 2D input device can be used to specify a 3D task only because of the use of the 3D model representation of the robot three-fingered hand. A three-dimensional world is different from a two-dimensional world because of its additional dimension. Using a 2D input device, the user needs to specify a 3D coordinate of X, Y, and Z. The user cannot possibly do so without extra help such as VRML. Using VRML to model an environment, given a 2D location in the model, a 3D coordinate is returned. This can be done because VRML makes use of perspective projection to transform points of a 3D object to the 2D-viewing plane; thus, it can reverse the operation from 2D back to 3D. This transformation is unlike parallel projection in that it preserves the depth information of objects in the scene, even after it has been transformed to 2D [Harrington].

The following are examples of perspective projection and parallel projection of the same object in the same 3D scene. A cube is projected to the viewing plane of $z = 0$. As one can observe from the unparalleled edges of the cube, through perspective projection, the depth information of the cube is preserved.

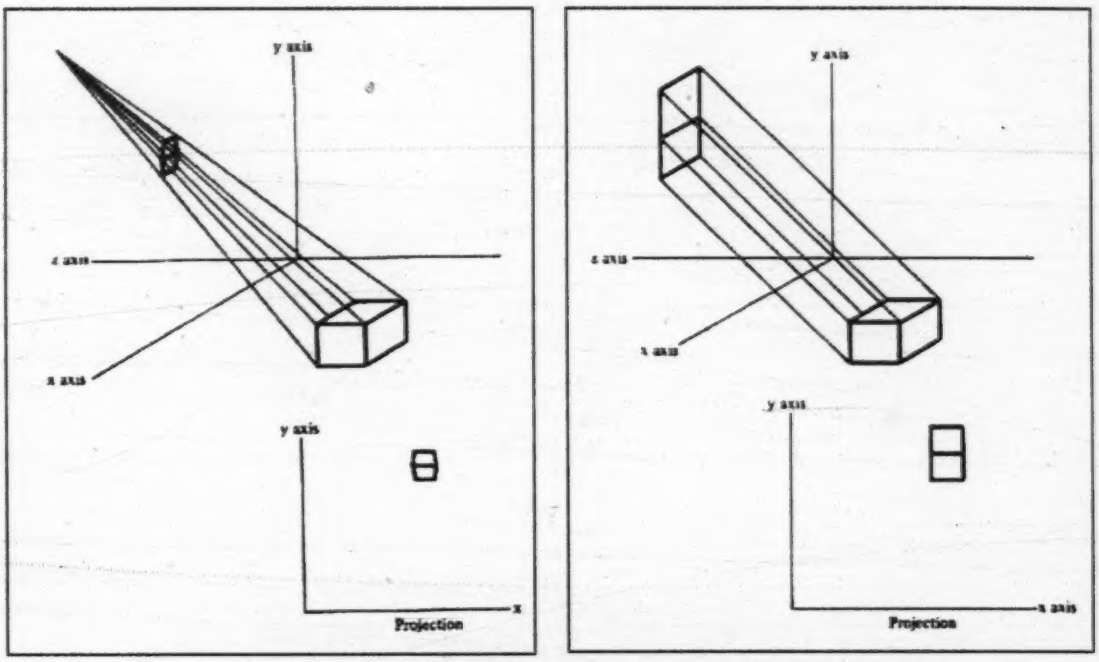

Figure 5. Example of perspective and parallel projections respectively.

Overall, VRML provides a great temporary solution to creating a three-dimensional representation of a remote robot and its environment by providing built-in solution for approximating 2D input to 3D representation.

4.4 Architecture of the Three-fingered Hand System

In our research, a sample teleoperation system is built based on the generic system architecture, as described in Chapter 3, to control a three-fingered hand (see Figure 6). This sample system is built using Java as the primary programming language. Thus, the resulting system is platform independent and object-oriented. Since the system is object-

oriented and modular, the components within the system can be interchanged easily. In each of the following sections, details about the implementation of each component in the sample system will be discussed.

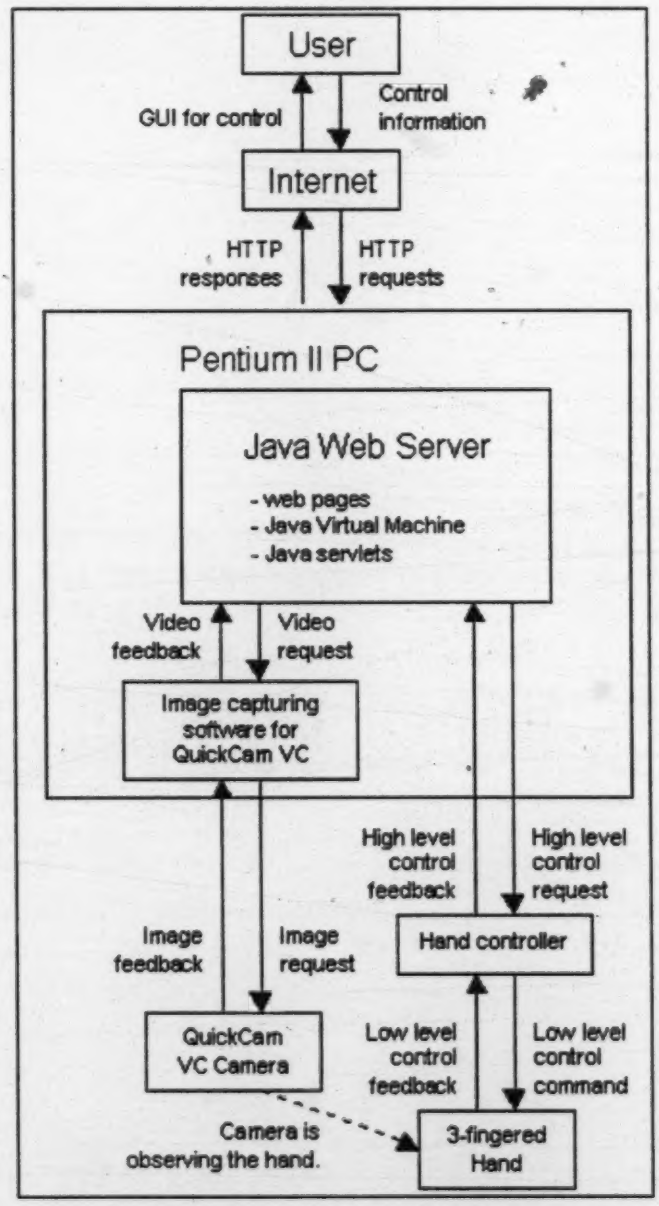

Figure 6. System layout of the three-fingered hand system.

4.4.1 Control Interface for Human Operator

The user of the three-fingered hand system is given a web-based interface. Through this interface, the user can move the hand freely or control it to move a "box". The interface is downloaded from the remote host and viewed through any web browser that is Java JDK 1.1 compatible and contains the VRML plug-in. The interface is composed of an live-image display on one side and a graphical model of the three-fingered hand on the other side (see Figure 8). This set-up allows a user to view the environment through video and then drag and move the hand via the model to control its movement. The hand can be controlled at three different levels: joint-level, finger-level and object-level. By using click-and-drag techniques on the VRML representation of the hand, the human operator can specify each finger joint's next position. Also, all three fingers can be moved as a group when the user specifies the position of a "box" through a set of predefined Java-based buttons. As the box is relocated, the end points of each finger tip is defined indirectly and fingers are moved accordingly.

4.4.1.1 Login

In order to allow only one person to control the robot at a time, a login system is implemented. This login system is created using Java and its Servlets technologies that allow a remote user to create a file-based database on the web server. Such database contains all users' email addresses and passwords. Because it is created and accessed through the use of Servlets, security measures are minimal and they are carried out automatically by the web server.

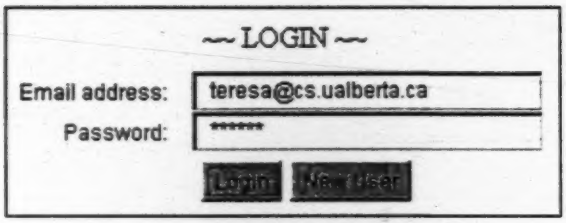

Figure 7. Snapshot of the login screen.

To create a new user, enter the user's email address and a password, then press the "New User" button. If the email address has been used already to register, it will return an error. If the New User is created successfully and the system is not currently in use, it will bring the user to the Main Console and he or she can control the hand for 5 minutes. Likewise, if a user has previously been registered, he or she simply needs to login by entering an email address and a password and then pressing the "Login" button. If the login is successful and the system is not currently in use, it will bring the user to the Main Console and he or she can control the hand for 5 minutes as well. If a "Busy" message is displayed, it means that another user is currently controlling the hand. The system will bring the user to the live-video display of the hand and ask the user to try to login again in 5 minutes. A new user can be successfully added but receive a "Busy" message.

In future implementations, all users should be verified through email to ensure that the user's email is valid and thus, they can be tracked when malicious actions are committed. For example, the user may try to crash the hand by crossing over the fingers for an extended period of time and that will cause damage to the servomotors.

4.4.1.2 Main Console

In the main console, there is a live-video display of the three-fingered hand on the left. On the right, there is a VRML-based three-dimensional representation of the hand itself. On the bottom left is a Java-based controller. It contains a set of pre-defined buttons that can be used to reposition the hand in place of the VRML model. On the bottom right is a text-based status display that shows information about the current user and any system messages.

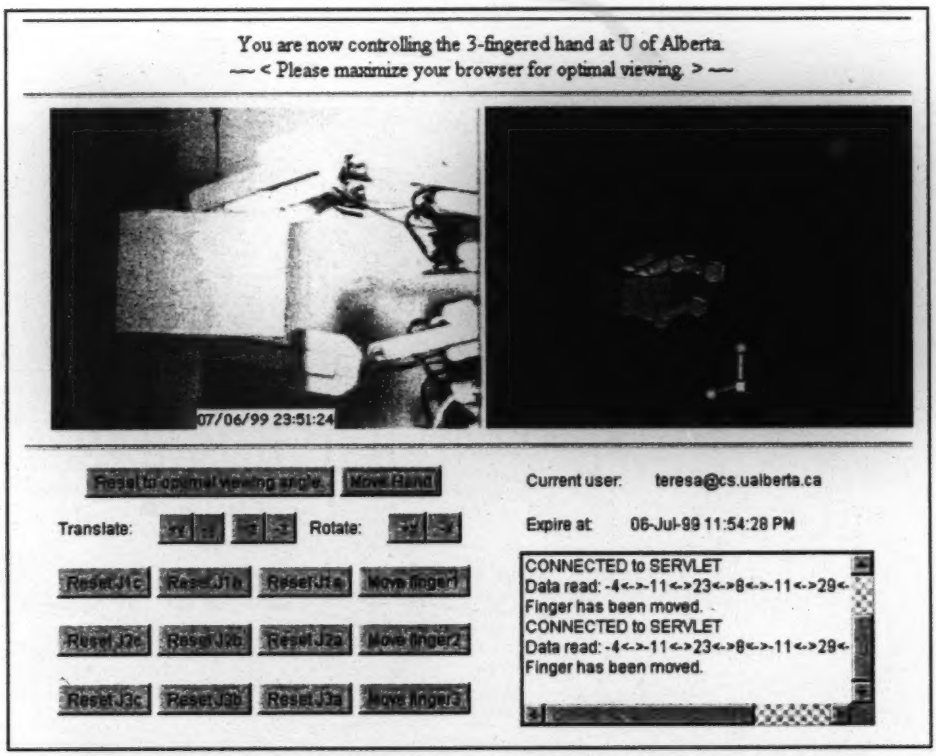

Figure 8. A snapshot of the control interface of the three-fingered hand system.

In this research, the robot, a three-fingered hand, is used because it can manipulate a rectangular-prism in 3D space. This 3D task can be specified by the human operator through the control interface. To control the hand itself, the VRML model can

be used. Each joint of the three fingers can be moved by click-and-drag. Also, a special joystick is created to change the view-angle of the hand. The view-angle can be changed as if the hand is rotated along the x and y axis.

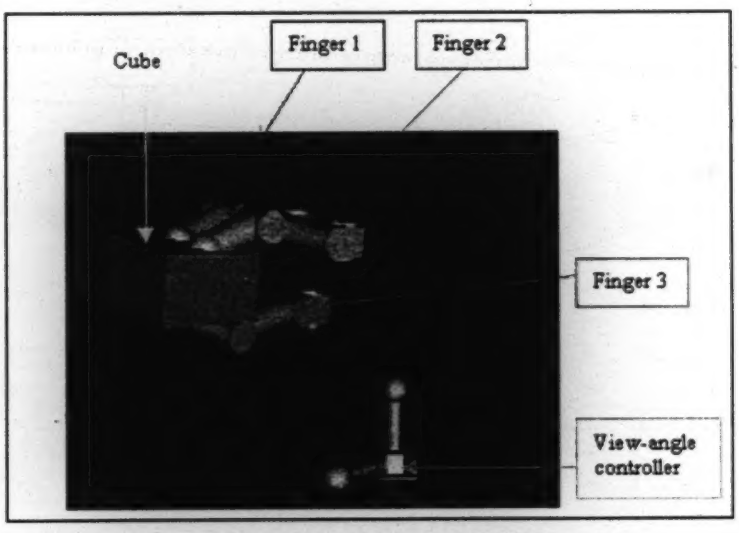

Figure 9. Labels of all the objects in the VRML model.

From the above diagram, it clearly shows that the 3D VRML model reflects the remote environment. It models the three fingers, the rectangular-prism that is being manipulated, and the virtual view-angle controller. Also, each joint's movement is limited. The following picture shows the range of rotation allowed on finger joint 2a:

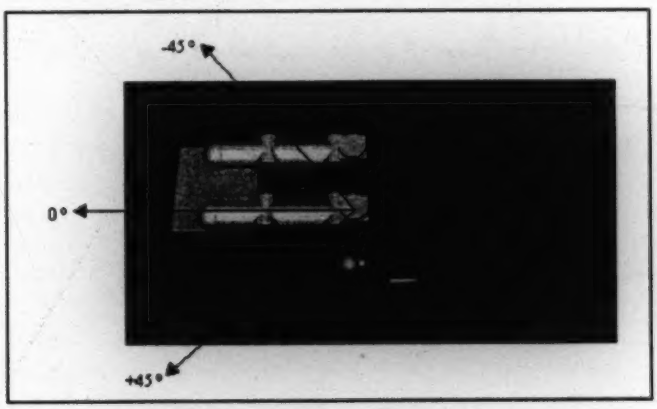

Figure 10. Limit of finger 2, joint a.

The following picture shows the angle of rotation allowed on finger joint 2b:

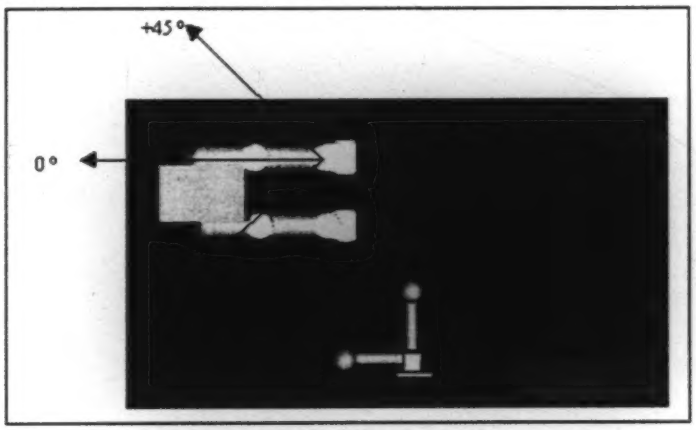

Figure 11. Limit of finger 2, joint b.

Finger joint 2c (the joint closest to the fingertip) has the similar limit as joint 2b, except that it cannot bend up from 0° to +45 °. Joint 2c can only bend down. Finger 1 and Finger 3 have the same joint limits as that of Finger 2. These limitations closely resemble those of a real human finger.

4.4.1.3 Virtual View-angle Controller

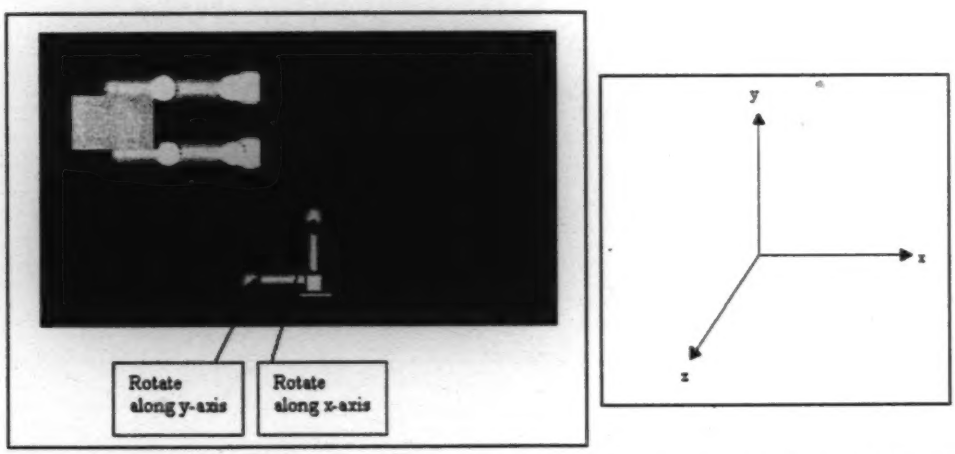

Figure 12. Shows the two shift nobs of the view-angle controller and the x, y and z axis in VRML.

A virtual controller is created to allow the users to manipulate the viewing direction of the 3D hand. It is in the form of a joystick and it has two shift knobs. One changes the viewing direction along the x-axis and the other changes the viewing direction along the y-axis. The view-angle can be changed 180 degrees either way. To change the view-angle, simply drag-and-move the different shift knobs. By rotating the view-angle, one can observe the rectangular-prism and the hand at different angles for precise control of both the rectangular-prism and the hand itself. For example, if one is concerned about how accurately the rectangular-prism is being moved, one may want to choose a view-angle that best shows the rectangular-prism. On the other hand, if one is concerned about the finger joints' movement, one may choose a view-angle that shows the most finger joints and not much of the rectangular-prism itself.

4.4.1.4 Quick Introduction to Click-and-drag in VRML

Once the user moves the mouse onto the VRML model, it changes the cursor's outlook. To control each finger joint, simply move the mouse over the desired joint. When the user sees the cursor turn into a "circle", the user simply needs to click-and-drag the finger joint accordingly. Then let go of the mouse button to set the joint to the new location. If that location is not desirable, simply click-and-drag again. All finger joints are movable through this method. Likewise, the view-angle controller can be controlled in this manner. However, after click-and-drag of the finger joints, it requires the user to press another Java-based predefined button in order to trigger the physical finger joint to be moved accordingly. The reason why changes in the VRML model do not immediately trigger an action is because users should be allowed to freely examine the next joint angles before committing to any actual movement of the hand.

4.4.1.5 Java-based controller

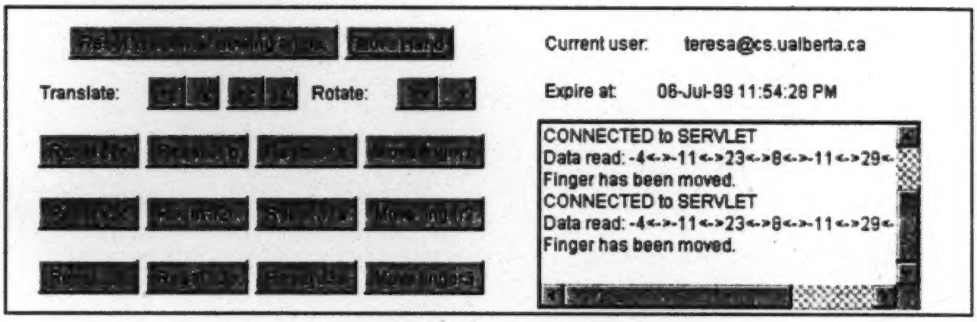

Figure 13. Snapshot of the Java-based controller.

The Java-based controller consists of a set of pre-defined buttons and a text-based status display of the hand. These buttons are used to complement the 3D model controller. The rectangular-prism is controlled through these buttons because to click-and-drag in the 3d model requires supreme precision by the human operator and the Java-based controller lessen the operator's work load by converting them into pre-defined buttons. Notice that no text-based input of coordinates is required; thus, the controller is user friendly and stress-free to the operator.

The Java-based controller contains a number of pre-defined buttons. A button, named "Reset to optimal viewing", is added to allow users to reset back to the optimal view-angle that is calculated when the console is first presented. At the optimal viewing direction, the user can see the most areas of the rectangular-prism; given the 3 fingers that are in the picture. The reason for having this button is that after the user manipulates the view-angle using the view-angle controller that is in the VRML model, it is best to provide the user the option of a one-step button to reset the view-angle back to optimal. Another button called "Move Hand" issues command to the host machine that will move the hand physically to match the current placement of the VRML model of the hand.

This button is created because manipulation of the 3D model does not trigger the real hand to be moved accordingly. It is necessary to add this button to trigger such action. A set of buttons called "Translate: +y -y +z -z" moves the rectangular-prism along the y and z axis in the VRML model. As it moves the rectangular-prism, the real hand will be moved accordingly as well. There is a limit as to how far the hand can move in each direction. Error messages will be displayed if the new position of the rectangular-prism is not reachable. Translations can only be done along y and z is because the joint c on all fingers have a limit of 0^{o} to 45^{o} and any translate along the x axis will require the joint c to exceed the limit. It is because the initial joint angle of joint c is at the extreme of close to 45 degree already and any change will exceed it. To rotate the rectangular-prism along the x-axis which means rotation of the rectangular-prism itself, users can use the buttons labeled "Rotate +x -x". As it moves the rectangular-prism, the real hand will be moved accordingly as well. There is a limit as to how far the fingers can tilt to each side. Error messages will be displayed if the new position of the rectangular-prism is not reachable. Rotation along y and z cannot be done due to the above reason also. The initial joint angles of joint c on all fingers are at one extreme and any change will exceed joint limit. Users can also resets the different finger joints to initial positions in the VRML model as shown in the following diagram by using the buttons titled "Reset Jxx". As it moves each finger joint, the real hand will be moved accordingly as well. It provides a shortcut for the user to set the joints back to initial positions.

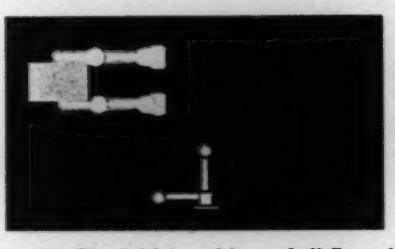

Figure 14. The initial positions of all fingerjoints.

The buttons labeled "FingerX" set fingerX's tip onto the rectangular-prism. It can be used after each finger joint was manipulated to a position that the fingertip is no longer on the rectangular-prism. It moves the real finger also. It triggers the calculation of inverse kinematics to reposition the finger joints accordingly onto the rectangular-prism.

All of the above pre-defined buttons allows the users to trigger a set of pre-defined actions. They provides shortcuts and serves as a supplement to the 3D controller. These buttons are created using Java and they can be viewed and used using a web browser that is Java JDK 1.1 comptiable.

4.4.1.6 Status Displays

From Figure 13, one can observe on the right hand side of the Java-based controller that there is a set of status displays. It shows the current user's email address and the expiry time of the current session. Also, there is a text box that shows any error messages and system status. These statuses have to be known to users at all time in case of errors and these text displays will keep users informed.

4.4.2 Java Servlet-Based Communication

Human operators use the Internet to connect to the remote web server and download the control interface. The interface then sends HTTP requests to the web server to issue commands to control the robot and feedback from the environment is transfered from the web server through the Internet to the users. Conventionally, CGI scripts are set up at the remote host to issue commands to the robot controller and to request for feedback information from the feedback controller. CGI scripts are mainly written in Perl or C and both these languages are platform-dependent. Also, CGI scripts are temporary such that they have to be loaded and executed by the web server every time an HTTP request is received. On the other hand, a Java servlet is loaded once by the web server and extra threads are spawned to handle the different simultaneous HTTP requests of the servlets. Java servlets save time in comparison with CGI scripts because of this loading overhead. Since Java servlets are written in Java, this allows the servlets to be object oriented as well. Object oriented components are easy to debug and extend when necessary.

Java servlets are similar to Java applets that run on the web server [Moss]. A user can send an HTTP request through the web browser to the web server. Then, the web server loads the requested Java servlet and runs it on the Java Virtual Machine that resides on the PC where the web server is located. The servlets communicate with the robot controller and send any responses in return back to the web server. The server then delivers feedback to the user through the web browser as shown in the following diagram.

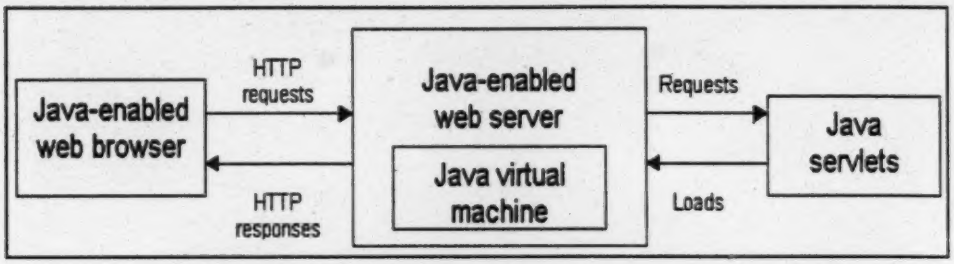

Figure 15. Flow diagram of how Java Servlets are served to a user through the web browser.

Since servlets are located at the web server and are executed upon receiving a request from the user, it is secure. Users cannot create their own malicious servlets to attempt a breach of security because all servlets reside on the web server. Furthermore, the web server will perform security checks during the execution of any serlvets. The previously mentioned server security makes it easier to implement servlets over CGI programs to establish network connections between the users and the remote host machine.

4.4.3 Remote Host Machine

The remote host machine in our sample system is a Pentium II personal computer (PC) with NT Workstation as the operating system. A Java Web Server [Sun], at http://rochfort.cs.ualberta.ca, is running on the host machine 24 hours a day; 7 days a week to service incoming HTTP requests. The control interface is also stored on the host machine and it is delivered to users as the web server receives requests. Furthermore, the remote host is connected to the three-fingered hand's controller and the video controller. The reason for choosing this system set-up as a remote host machine is because of its low cost and its ease of use. For example, the remote host machine can be easily connected to the robot through a serial cable. Furthermore, the video camera is

plugged into the computer directly through its USB port. The use of Windows NT as the operating system also provides a great platform to run the different software on. The Sun Java Web Server can be easily installed and run on Windows NT. Overall, the remote host machine is chosen to be a Windows-based PC because of convenience.

4.4.4 Robot and Its Controller

A three-fingered hand with 9 degrees of freedom is used as the slave robot. The task to be performed is to move the fingers freely and to hold onto a rectangular-prism. The hand is composed of a stand and a mount for all three fingers. The links between all finger joints and the handstand is made with wooden sticks. The nine finger joints are made of servomotors CS-21 BB from CIRRUS. Each servo is approximately C$50.

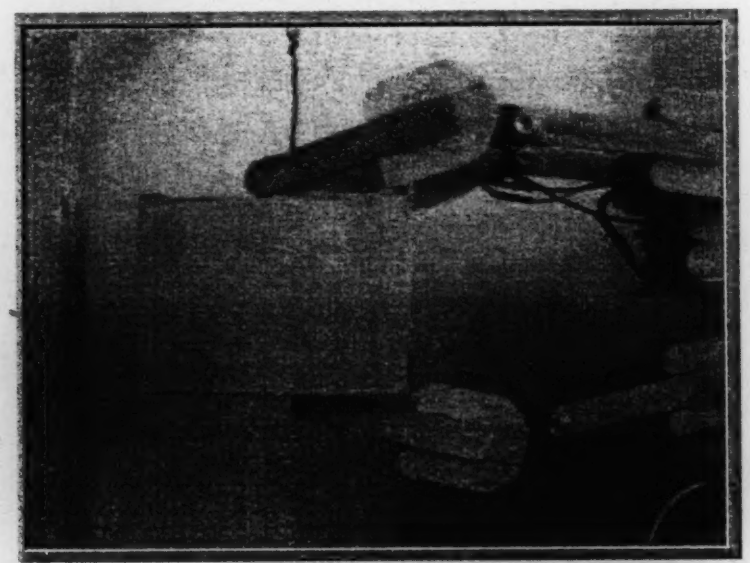

Figure 16. Picture of the three-fingered hand.

To control the finger joints, two Serial Servo Controllers (SSC) for R/C servos are used. These controllers use a computer's serial port to control up to eight servos on each

controller. The two SSCs are hooked in parallel and they accepts serial input at 2400 bps and capable of outputting a total of 16 steady channels of servo-control signals. These SSCs are made by Scott Edwards Electronics, Inc. Each controller is approximately C$70. There are many sizes of servomotors but in order to fit on each of the finger joint, the type of servos used are sub-micro in size and as indicated with a white arrow in the following picture.

Figure 17. Picture of servomotors of different sizes.

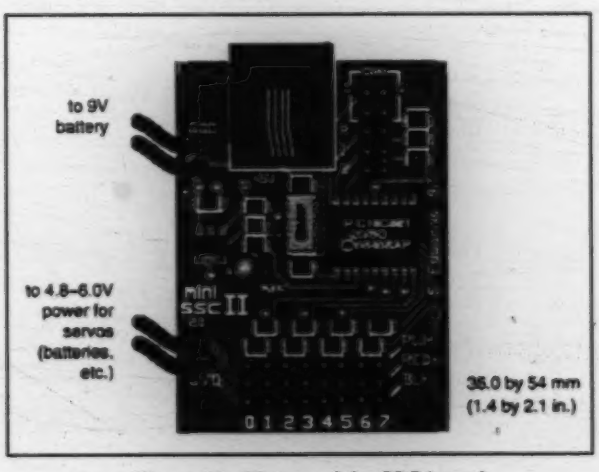

Figure 18. Picture of the SSC board.

The above picture shows one of the two Serial Servo Controllers used to send signal to the nine finger joints. Two of these controllers are joined in parallel and hooked to the remote host and the robot hand as illustrated in the following diagram.

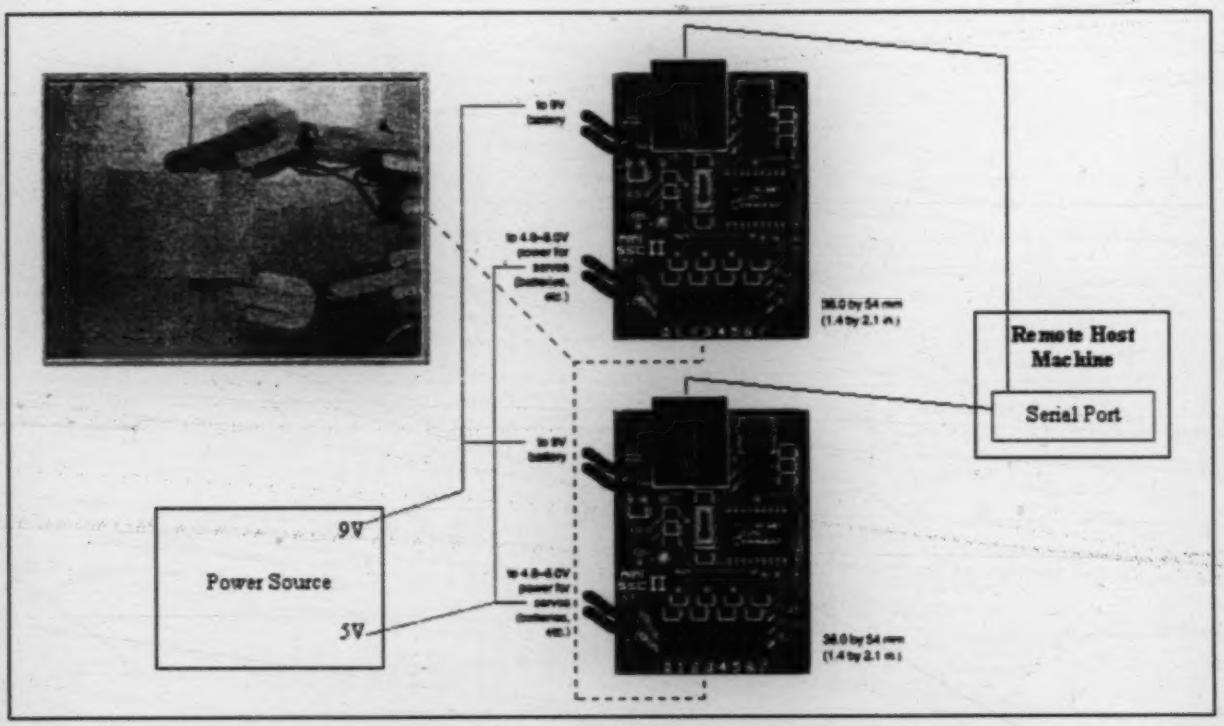

Figure 19. Diagram of the two SSCs, the hand and the remote host machine.

At the remote host machine, there is a Java-based controller that waits for users to send manipulation instructions through Java-servlets. These instructions are gathered and convert to motor positions by this software controller and then send through the serial port to the SSCs. Finally, SSCs output appropriate servo-control signals to the nine servomotors and thus, all three fingers of the hand would be relocated accordingly.

4.4.5 Sensory Feedback

Telerobot provides the operator with a representation of the remote environment through a live video display. In teleoperation, the operator must make frequent observations to ensure that actions occur as intended; otherwise, the operator needs to revise the control instructions that are being sent.

Through the video, one can observe the hand's movements. It is a Java-based display. Thus, you need a Java-based web browser in order to see it run properly. Netscape v 4.61 is recommended. On the bottom of the video display, it shows the current local time. Through this clock, you can also observe the delay in transmitting the video. Each video image is 320x240 pixels and 16 bits in color. The video delay varies and it is dependent on the network, the workload of the host machine, the system set-up of the user's machine and its Internet connection.

A QuickCam VC [QuickCam] is used as the video camera to observe the environment. QuickCam VC was chosen because of its low cost and adaptability to any PC by simple plug-and-play into either a parallel port or USB port. No extra image capturing hardware is required. The image capturing software is Java-based and it runs on the remote host machine. It captures images, which are 352 X 288 pixels (in Common Interchange Format (CIF)) and in 16-bit colors, from the camera consistently and stores the images in predefined locations on the remote host. QuickCam VC can capture up to 15 frames per second at CIF. A Java applet on the control interface requests for captured image to be displayed by sending HTTP requests to the web server; as the web server receives the request, it locates the next image to be displayed at the predefined location of the host machine. The following diagram shows a picture of the above transaction. In

this manner, images are captured automatically and they are ready to be transferred at the time of request. The delay of the image delivery is minimized and it resulted in delivering one quality image per second. Besides the lengthy transfer of the image data, the bandwidth of the Internet is limited and inconsistent; video feedback is therefore unreliable as the sole source of feedback to indicate the status of the robot and the environment.

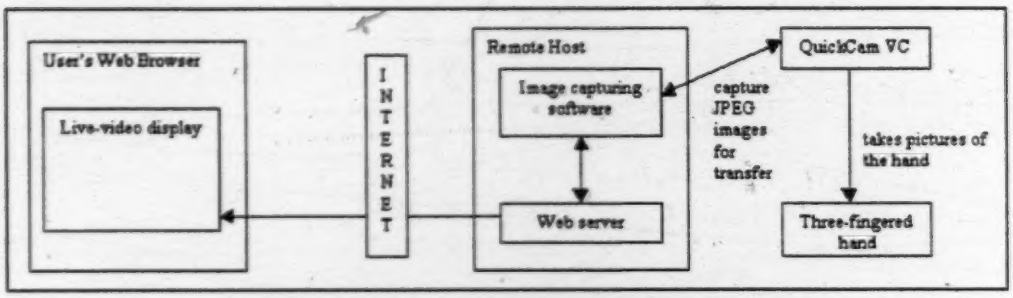

Figure 20. The overall picture of all imaging components.

An alternative feedback is the current kinematics positions of the robot. The positions can be simply a set of numeric values and the transfer of such data is effortless compared to video transfer. To make the positioning data easier for the user to comprehend, a graphical model of the robotic operator is constructed and displayed at the user's end. This graphical model serves as a display of the current status and also an interface for the control of the next movement of the operator as shown by the three-fingered hand system. This form of graphical control requires minimal technical details from the user; yet it provides the user easy control of the robotic operator. As a user moves the graphical model of the hand, the robot is moved at the same time and any inconsistency between the real robot and the model's displacement will be corrected as the robot provides feedback to the model.

Using such VRML models to serve as both feedback and input (as described in previous section) to users of an IBTS is the first of its kind. Such 3D models improved the usability of the overall system by providing the users more a more accurate and up-to-date representation of the remote location in comparison to the live-video display. Also, this form of new input format is very easy to use even for those that are not familiar with the control of a robot.

4.5 Optimal View-angle

Now that we have finished examining each component of the three-fingered hand IBTS, we can look more closely into the steps in finding an optimal viewing direction. In order to find the optimal view-angle, we have to first define what an optimal view-angle is; then, we have to study the graphical concepts of three-dimensional rotations, plane equations, back face elimination and z-buffering. Plane equations need to be found to represent each surface of the objects in the scene. Back face elimination is used to remove surfaces that are not facing the viewers. Finally, z-buffering will be applied to determine which surfaces are closest and most visible to the viewers. These graphical concepts are important in finding the optimal view-angle of the object that is being manipulated by the three-fingered hand and they will be studied in details in the following sections.

4.5.1 Defining the Optimal View-angle of the Manipulated Object

The optimal view-angle is when the object that is being manipulated, in this case the rectangular-prism, is the most visible. Besides the rectangular-prism, there are other objects in the scene. All the fingers and the hand stand have to be accounted for when

searching for the optimal view-angle. Besides the visibility of the surface, each visible surface should also be approximately equal in portions to create a balanced view. For example, if two of the surfaces of the prism are visible when the view-angle is rotated along x-axis by *u* degrees, then we have to find out how much of the two surfaces is visible. If one surface is around 70% visible to the viewer but the other surface is only 10% visible, then this view-angle is not desirable due to the average deviation of the visible surface is large. On the other hand, if a view-angle that is rotated along x-axis by *v* degrees, there are two surfaces visible and they are both 50% visible, then this view-angle will be more preferable than the previous one because both surfaces are half visible to the viewer. However, if a view-angle that is rotated along x-axis by *w* degrees and there are 3 visible surfaces, this view-angle will be more desirable despite the portions of the surfaces that are visible. This is because it is better to view a bit of an extra surface then to view no part of that extra surface.

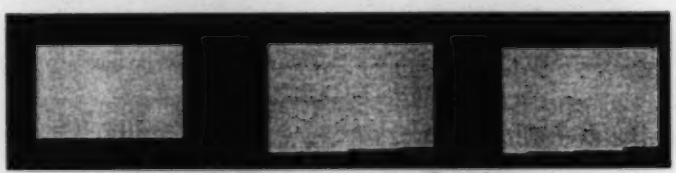

Figure 21. Three view-angles of a prism.

The above diagram shows three different view-angles of a rectangular prism. From the first shape on the left, the viewer cannot even tell that it is of three dimensions but by viewing a bit of the left surface as shown by the middle drawing, the viewer can at least tell that it is not flat. Then from the third drawing, the viewer can tell that the shape resembles a prism. It does not matter that only small portions of the additional two

surfaces are visible; they are at least visible to the viewer and provide users with depth information.

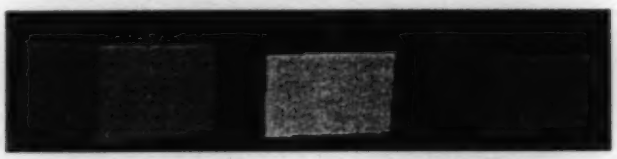

Figure 22. Different view-angles show different portions of the 3 visible surfaces.

Also, the above diagram presents three different view-angles that show three surfaces of the prism. From the drawing on the right side, the viewer obtains the best out of the three view-angles because it shows three surfaces that they are the closest in areas; thus creating a balanced view of the prism. This sense of balance is important. In a recent survey conduct by NASA, it is pointed out that when using virtual environment technology, if a sense of balance is maintained, the resulted virtual environment is more user-friendly [Kenney]. Overall, the optimal viewing angle can be expressed as followed.

Given visible areas A_m and m is the number of visible surfaces, o is the optimal view angle if and only if at o, m is at its maximum and given the same m, the difference, *Diff*, between the areas of the visible surfaces is at its minimum.

$$Diff = \frac{\left| \sum_{j=1}^{m} \left(\frac{A_j - \frac{\sum_{i=1}^{m} A_i}{m}}{\frac{\sum_{i=1}^{m} A_i}{m}} \right) \right|}{m}$$

For example, if at view angle V_1, max(m) is 3 and the visible areas A_m are 380, 228, and 247 units each, then using the above formula, *Diff* is calculated to be 0.22. If at view angle V_2, three areas are visible and the areas are 380, 304, and 152 units each, then *Diff* = 0.30. Given view angles V_1 and V_2, V_1 would be preferred because it yields a

smaller *Diff* value. The goal is to find a view-angle that yields the most number of visible areas of the manipulated object and also maintain the smallest deviation between the visible areas and thus, the view-angle should give the smallest *Diff* value.

4.5.2 Rotation in 3D Space

Rotation in three dimensions is a basic geometric transformation in computer graphics. Since the objects in the 3D space may be rotated, it is important to understand how it is done. Rotation transformation is defined as a rotation about the origin of the coordinate system by a specified angle θ. First, we will understand transformation in two dimensions. In the following diagram, it shows a point *P(x,y)* rotated by θ and transformed to point *P*(x,y)*. The dashed line indicates the change from x to x*.

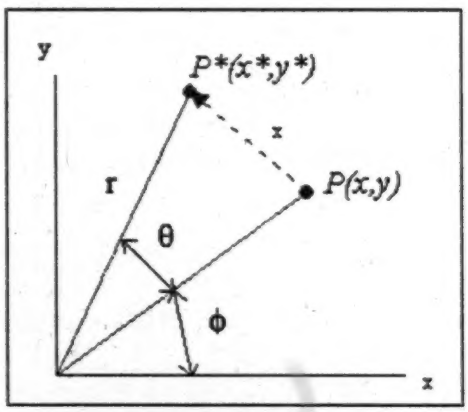

Figure 23. Rotation of a point.

Given *P(x,y)* in Figure 23, *P*(x*,y*)* can be calculated using simple trigonometric relations.

$\cos\phi = x / r \quad \Rightarrow \quad x = r\cos\phi$

$\sin\phi = y / r \quad \Rightarrow \quad y = r\sin\phi$

$$\cos(\phi+\theta) = x^*/r \quad \Rightarrow \quad x^* = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$$

$$\sin(\phi+\theta) = y^*/r \quad \Rightarrow \quad y^* = r\sin(\phi+\theta) = r\sin\phi\cos\theta + r\cos\phi\sin\theta$$

Then substitute x and y for their respective values:

$$x^* = x\cos\theta - y\sin\theta$$

$$y^* = x\sin\theta + y\cos\theta$$

or in matrix form:

$$[x^* \; y^* \; 1] = [x \; y \; 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The above 2D rotation is along the z-axis and in order to expand it to rotation along z-axis in a 3D space. The following matrix is derived [Anand]:

$$[T_R]^{\theta}{}_Z = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and it maps to

$$x^* = x\cos\theta - y\sin\theta$$

$$y^* = x\sin\theta + y\sin\theta$$

$$z^* = z$$

Similarly, rotation along x-axis by θ can be obtained using the following equations.

$$x^* = x$$

$$y^* = y\cos\theta - z\sin\theta$$

$$z^* = -y\sin\theta + z\cos\theta$$

or

$$[T_R]^{\theta}{}_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotate along y-axis by θ can be expressed as

$$x^* = x\cos\theta + z\sin\theta$$

$$y^* = y$$

$$z^* = -x\sin\theta + z\sin\theta$$

or

$$[T_R]^{\theta}{}_Y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4.5.3 Equation for a Plane Surface

An equation for a plane surface can be expressed in the form [Hearn]:

$$Ax + By + Cz + D = 0$$

where (x,y,z) is any point on the plane. To find *A*, *B*, *C*, and *D*, we need three vertices on the plane's boundary and we will denote these points as (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3). We first try to solve the following set of simultaneous plane equations for the ratios *A/D*, *B/D*, and *C/D*:

$$(A/D)x_i + (B/D)y_i + (C/D)z_i = -1;\ i = 1, 2, 3$$

If we use Cramer's rule, we can express solution for *A, B, C, D* in the following determinant form:

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix}$$

$$B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$C = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix}$$

$$D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

If we express the above in explicit form, we get the following:

$$A = y_1*(z_2-z_3)+y_2*(z_3-z_1)+y_3*(z_1-z_2)$$

$$B = z_1*(x_2-x_3)+z_2*(x_3-x_1)+z_3*(x_1-x_2)$$

$$C = x_1*(y_2-y_3)+x_2*(y_3-y_1)+x_3*(y_1-y_2)$$

$$D = -x_1*(y_2*z_3-y_3*z_2)-x_2*(y_3*z_1-y_1*z_3)-x_3*(y_1*z_2-y_2*z_1)$$

For example, a surface with vertices (2,0,0), (2,1,0), and (0,1,0) as shown in Figure 25, can be expressed as

$$Ax + By + Cz + D = 0$$

where $A = 0*(0-0)+1*(0-0)+1*(0-0) = 0$

$B = 0*(2-0)+0*(0-2)+0*(2-2) = 0$

$C = 2*(1-1)+2*(1-0)+0*(0-1) = 2$

$D = -2*(1*0-1*0)-2*(0*0-0*0)-0*(0*0-1*0) = 0$

Therefore, $z = 0$ can be used to express the polygon given those three vertices.

4.5.4 Back Face Elimination

A back face is a surface that is turned away from the viewer. When the 3D model represents a solid object, these back faces are not visible and thus, they are skipped in the drawing process [Hill]. In this thesis, VRML is used to render the 3D model of the hand but it fails to provide a function that returns whether a surface is visible or not. Thus, extra calculations are required to determine back faces. Back face elimination is sometimes called hidden surface removal also. By removing these hidden surfaces in the scene, it gives a smaller number of surfaces to deal with during z-buffering. In the case of back face elimination, given a rectangular-prism that is positioned as in the following diagram, only surfaces A, B and C will not be removed; the other three faces of the prism will be eliminated.

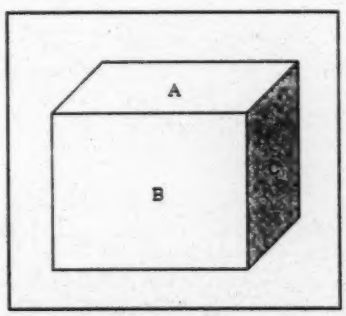

Figure 24. Surface A, B and C are not removed by back face elimination.

To perform back face elimination of a cube with six surfaces, we need to find the normal vector of the projected area of each surface polygon by using the following formulas.

If the n vertices of a surface polygon are (x_i, y_i, z_i), then:

$$A = \sum(y_i - y_j)(z_i + z_j) \quad \text{for i = 1 to n.}$$

$$B = \sum (z_i - z_j)(x_i + x_j) \quad \text{for i = 1 to n.}$$

$$C = \sum (x_i - x_j)(y_i + y_j) \quad \text{for i = 1 to n.}$$

If i = n, then j = 1; otherwise, j = i + 1.

The result [*A B C*] is a vector normal to the polygon. If *C* is positive, then the polygon faces towards the viewer. This is because by comparing the directions of two vectors, the vector normal to the polygon and the depth direction vector [0 0 1] , if they are the same, then the polygon is facing the viewer. To compare the directions of two vectors, we have to compute their dot product, *dp*.

$$dp = [A\ B\ C] * [0\ 0\ 1]$$

$$= A*0 + B*0 + C*1$$

$$= C$$

The dot product is actually equal to *C*. The dot product of these two vectors gives the product of the lengths of the two vectors times the cosine of the angle between them. If the angle between the two vectors is between 0° and 90°, then the cosine is positive and thus, the dot product will also be positive. If these two vectors are opposing, then the dot product is negative. In other words, the polygon is not facing the viewer. Since *C* is the dot product of the vectors, if *C* is positive then the polygon is facing the viewer; otherwise the polygon is a back face.

The above set of formulas is shown by Harrington and it is useful because it not only yields the normal vector of the polygon, it also dervives the projected area of the polygon. For example, the resulted *C* value represents two times the area of projection of the polygon on the xy plane. This area of projection value is useful when debugging the algorithm to determine which surface is visible to the viewer. Since the viewer is looking

at the projection of the polygon on the xy plane, the *C* value must not be zero. The following example will show this overall concept of back face elimination more clearly.

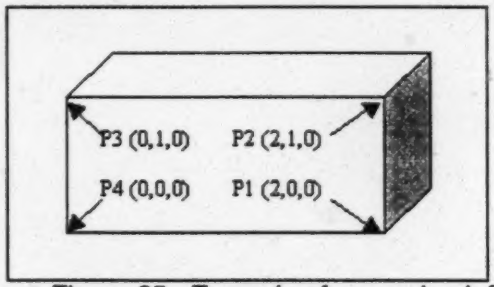

Figure 25. Example of a non-back face.

For example, from the above figure, we are given the four points of one surface of the cube. We can find out if this surface is facing the viewer if we find out its *C* value.

$$C = (2-2)(0+1)+(2-0)(1+1)+(0-0)(1+0)+(0-2)(0+0) = 4.$$

Since 4 is positive, we can conclude that this surface of the cube is facing the viewer. Also, the *C* value of 4 indictated that the area of projection of the polygon on the xy plane is of 2 units; this reassures that the polygon is visible to the viewer. However, the four points of the surface in the Figure 26 will return a negative *C* value of

$$C = (0-0)(0+1)+(0-2)(1+1)+(2-2)(1+0)+(2-0)(0+0)= -4.$$

This negative *C* value shows that the surface outlined in Figure 26 is a back face and it will be elminated.

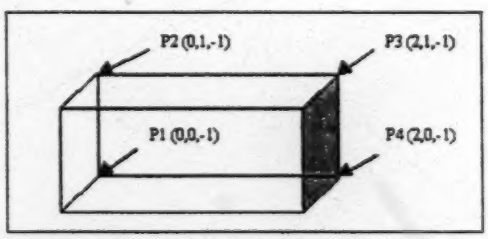

Figure 26. Example of a back face.

4.5.5 Z-buffering

Z-buffering is used to determine which part of an object is visible to the viewer. It uses an extra buffer that is the same size of the projected work space. This buffer is initialized to negative infinity. Then for each non-"back face", the z-value of the surface is compared with the stored value in the buffer. For example, if (x,y) is a point on a surface A, then find the z-value for that point; if the z-value is higher than the stored value in the z-buffer, then take note that this surface is visible at point (x,y) and stored its z-value into the z-buffer. If another surface, B, is visible at point (x,y) and it has a higher z-value than the surface A, then it will store surface B's z-value instead and take note that this z-value is associated with surface B. At each point (x,y), only one surface can be visible. At the end, the z-buffer will contain all the z-values of surfaces that are on the top level and thus, one can determine which surface of the rectangular-prism is visible given all the other elements in the scene. Also, from the z-buffer, it can be determined how much of each surface is visible. By repeatedly calculating the z-buffer given all possible degrees of rotation along x, y, and z-axis, an optimal view-angle can be found. The trade-off for such findings is that it is time and resource consuming. But overall, it serves its purpose in finding an optimal view-angle. The following diagram shows a number of overlapping shapes in a scene and the color parts of each shape represent the parts that are stored in the z-buffer and associated with the corresponding shape.

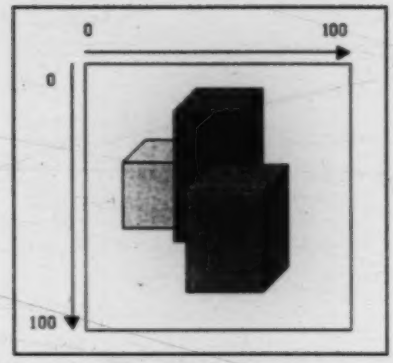

Figure 27. A z-buffer of size 100x100 and its content of the colored faces of each of the 3 shapes.

Furthermore, given the surfaces as indicated by the four points in Figure 25 and Figure 26, the surface in Figure 25 will be stored in the z-buffer and not the surface in Figure 26. The z-buffer will show that one surface is visible only and it is the surface as described in Figure 25 because the z-value of the surface in Figure 25 is higher than the corresponding value of the surface in Figure 26.

4.5.7 Finding the Optimal View-angle

Now that the optimal view-angle has been defined and all the basic graphical concepts have been explained, we can look into the specific steps in finding the view-angle that best shows the object that is being manipulated by the three-fingered hand.

Step 1. Find the 8 vertices of each object in the scene, given the center location, length, width and height of each object that is in the scene. These vertices will be used in Step 2 in performing back face elimination. Because objects in the scene may be rotated along x, y and/or z-axis, all vertices needed to be transformed accordingly as shown in the previous section.

Step 2. Perform back face elimination of all objects. By removing all the faces that are not visible to the operator, it simplifies the problem of determining how much of the manipulated object is visible.

Step 3. Find an equation in form of $Ax + By + Cz + D = 0$ to represent the projected plane of the each polygon faces that are not eliminated in Step 3.

Step 4. Using the equations found in Step 3, find the depth values for all position (x,y) of each plane. Then perform z-buffering to determine which polygon is being seen by the viewer and resulted in how much the rectangular-prism is being seen.

Step 5. Repeat Step 1 to 4 at different rotation angles to determine the optimal view-angles.

The above calculations are time-consuming. Given the three fingers and one rectangular-prism in the model, it takes three hours on a PC Pentium 100 to find the optimal view-angle. To speed up the calculations, the search algorithm can be changed to limit the search space to sections where the objects locate and ignore empty spaces in the environment. During the calculations, multiple optimal view-angles may be found and the first one found is taken to be the best one because it involves the less change from the initial view-angle. As a result, the optimal angle is found to be at an angle that is rotated $+29.80^{\circ}$ along x-axis and $+40.12^{\circ}$ along y-axis. The following diagram shows such a view-angle.

Figure 28. Optimal view-angle for the rectangular-prism.

As one can observe from the above diagram, the human operator can see three sides of the rectangular-prism and each side is mostly equal in surface areas. The following diagram shows a different view-angle of the hand. It is at an angle that is rotated at +29.80° along x-axis and +30.02° along y-axis; this is 10 ° less rotation along the y-axis compared to the optimal view-angle as shown in Figure 29. As a comparison, one can see that at +30.02°, less of the left side of the prism is visible to the viewer. In terms of numbers, the optimal view-angle has a *Diff* value of 0.17 but at +30.02°, the *Diff* value is 0.39. Thus, by calculating the *Diff* value, it shows that a little change in visible appearance to the viewer may be reflected significantly in the *Diff* value.

Figure 29. 10° from the optimal view-angle.

As one can predict, there may be more than one view-angle that has the same minimal *Diff* value. In this case, there are at least two other ones. But from the pictures below, one can understand why the previous view-angle is picked as the optimal instead.

Figure 30. Not-so-optimal view-angles.

It is to note that the two above view-angles both have the same *Diff* value as the optimal view-angle. The optimal view-angle is chosen over the above two angles because it is the first to be found and thus, the closest to the original view-angle and assumed to be the most friendly and logical view-angle.

With the current implementation, the optimal view-angle is found at the beginning of each control session only. Due to time-constraints, we did not optimize the search algorithm. In the future, these calculations should be sped-up. For every movement of the manipulated object, a new optimal view-angle should be found. That way, the human operator can always view the rectangular-prism at the optimal view-angle despite relocations of the rectangular-prism.

4.6 Robot Kinematics

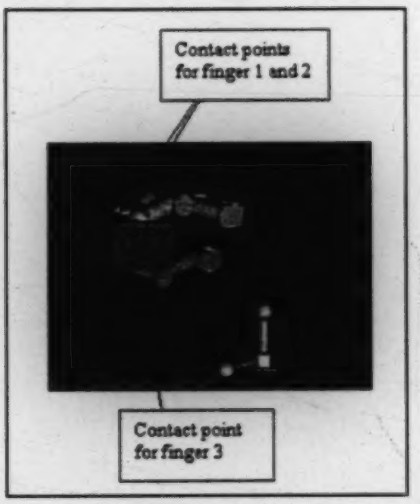

Figure 31. Contact points of the three fingers and the rectangular-prism.

If we want the hand to hold onto the rectangular-prism, we need to find the contact points of the three fingers and the rectangular-prism. This can be easily done because the VRML model keeps track of the center location of the rectangular-prism. Given the center location (x,y,z) and the dimensions (l, w, h) of the rectangular-prism, the three contact points can be found using simple arithmetic operations.

After finding the three contact points, we have to find the joint angles for each of the nine joints to position such that the three fingers are at the contact points and the hand will hold onto the rectangular-prism at a given location. To do so, we use inverse kinematics as described in the following section.

Kinematics is the study of motion without regard to the forces which cause it [Craig]. It has to do with the position, velocity and acceleration of an object. In this thesis, we are concerned especially with the position of each joint of the hand. Because there are nine finger joints and nine corresponding independent position variables that

would have to be specified in order to locate all parts of the hand, the three-fingered hand presented in this thesis can be described as having nine degrees of freedom.

Three joints are presented in each finger and high performance sub-micro servomotors (made by CIRRUS, model CS-21) are used to move these joints. The servos also serve as position sensors that allow the relative position of neighboring links to be measured; thus, each fingertip's location can be found [Craig]. Each fingertip is a free end of a chain of links that makes up one finger; so each fingertip can also be called an end-effector of the hand. When studying the kinematics of the hand, the position of each end-effector is important. Forward kinematics involves the problem of computing the position and orientation of the end-effector. For example, a set of joint angles will be given and the task is to find the position of the fingertip relative to the origin of the hand that joins the finger with the hand. This can be done using the following formulas as described by [Craig]:

$$P_x = \cos\theta_1 \, (L1^* \cos\theta_2 + L2^* \cos\theta_{23})$$

$$P_y = \sin\theta_1 \, (L1^* \cos\theta_2 + L2^* \cos\theta_{23})$$

$$P_z = - L1^* \sin\theta_2 - L2^* \sin\theta_{23}$$

The resulted (P_x, P_y, P_z) is the new location of the finger tip if θ_1, θ_2, and θ_2 are given as the next movement of the joints of the finger.

The diagram below shows the corresponding joint angles and lengths of the links between the angles as used in the formulas.

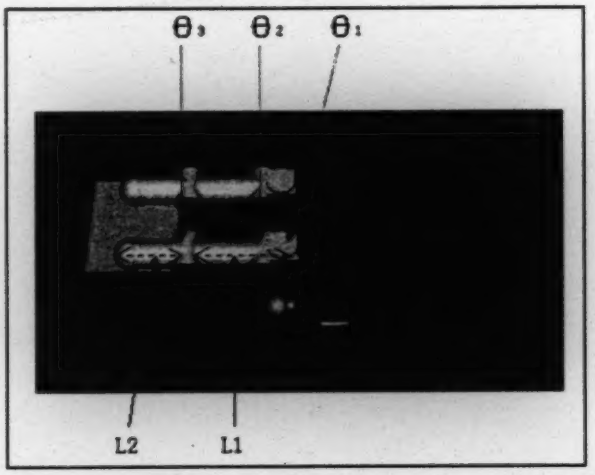

Figure 32. Labels of the joint angles and the lengths of the finger links.

In inverse kinematics, it is the reverse. A position for any one of the end-effectors may be given and the task is to find the joint angles for respective finger joints. Given a position for any one of the end-effectors, the following equations are used to find the joint angles: θ_1, θ_2,and θ_3 [Craig].

$$\theta_1 = \text{Atan2}\,(P_y, P_x)$$

$$\cos\theta_3 = (P_x^2 + P_y^2 + P_z^2 - L1^2 - L2^2) / (2 * L1 * L2)$$

$$\theta_3 = \text{Atan2}(\text{sqrt}(1-\cos^2\theta_3), \cos\theta_3)$$

$$\theta_2 = - \text{Atan2}(P_z, \text{sqrt}(P_x^2 + P_y^2)) - \text{Atan2}(L2 * \sin\theta_3, L1 * \cos\theta_3)$$

The resulted θ_1, θ_2,and θ_3 are the joints angles that have to be reached in order for the finger tip to be located at (P_x, P_y, P_z). The calculations of these joint angles are done at the client machine and it is embedded in the main console and tied closely to the VRML

model and the Java-based buttons. No explicit steps are required by the human operator; the calculations are automatically triggered as the operator move a rectangular-prism up, down, or side-to-side. For example, when the rectangular-prism moves up one unit, the new center of location is provided by VRML. Based on this center location, a point of contact for each finger can be found using Java because the prism's dimensions are known. After finding each point of contact, we can use the above formulas to find the three different joint angles for each finger. Finally, all nine finger joints will be moved accordingly to the new points of contact and thus moving the prism to its new location as dictated by the user.

Chapter 5

Evaluation Criteria

5.1 Introduction

Comparative studies are difficult to perform because this is the first IBTS that controls a three-fingered hand robot. However, like other IBTS, the three-fingered hand system can be judged based on some numeric evaluations. The following sections describe each of these evaluation criteria and shows how it is applied to the three-fingered hand IBTS. The Australian IBTS created by Talyor in 1995 is used as comparsion in some of the criteria. This Australian system is chosen because it is one of the first IBTSs and it is the so far the best documented one.

5.2 Video Quality and Speed

A video display is used in almost every IBTS because it provides continuous transfer of images from the remote host to the human operator through the Internet. This video display can be measured based on the size of the image, the color depth, and the compression that is used. In this research, the image is 352 x 288 pixels and 16 bits color. This image size is of the standard Common Interchange Format (CIF). 16 bits color is used over 24 bits color to reduce image size and yet maintain a colored-

perception of the remote environment. The resulted live-images are satisfactory in display size and color that they are pleasant to the eyes of the human operators. To reduce image size, JPEG compression is performed by the on-board image capturing capability of the camera. Overall, the live-video display provides 1 frame per 3 seconds to the end-user. This delay shows big gaps between movements of the hand; thus, it should be improved in the future. These 3-second delays are found to be in the Java-based capturing software that is located at the remote host. It takes one full second to capture the JPEG image from the camera to the remote host itself. Then, Internet delays the transfer the transfer of the image to the user by approximately half a second. Finally, the Java-based live-video display at the user's end takes 1.5 seconds to load and display the image in the web browser itself. At least 2.5 seconds in delay can be minimized as the programming language, Java, matures and speeds up in the future. Currently, Java is an interpretative programming language. After the Java-based image display program is downloaded to the client machine, an interpreter on the client machine, called the Java Virtual Machine, interprets the byte code of the program and converts it to machine-specific instructions for execution. This process is time-consuming compared to direct execution of machine code. However, this interpretive process is what gives Java its platform-independent capability. Newer versions of Java now promote Just-In-Time Compiling such that machine-specific byte code can be produced for direct execution. In comparison, the IBTS at University of Western Australia [Talyor, 1995] delivers video images that are 256 x 256 pixels in size and 8-bits in colors. It takes an average of two seconds per image. This is one second less than that of the three-fingered hand IBTS and it is because the image is smaller in size and it has half the number of colors available.

Also, as explained above, some extra delay in the three-fingered hand system is due to Java's interpretive nature.

5.3 Actual Execution Time of the Robot

The responsiveness of the control interface is important in an IBTS. It is undesirable when the human operator issues a command and it does not get executed in a timely fashion. In the three-fingered hand system, from the moment that a user issues an instruction from the interface to the time when the robot receives that instruction and carries out the execution, it takes less than a second. That is because an instruction is just made of a few numbers that specify which joints and what joint angles to move next. The conversion from the VRML model input and the Java-based input to these low-level joint-specific instructions is performed on the client machine and is fast because they are relatively simple arithmetically. There is almost no delay in execution of any instructions. On the other hand, the Australian system [Taylor, 1995] takes an average of 8 seconds to process a request and 15 seconds to transmit the request data to and from the user. It is due to the older web technologies, like Perl, that were used in 1995 and the robot itself. Use of Perl scripts to process request requires the extra step of data validation and user's security check. This extra step may be time consuming. Also, the robot used in the Australian system is a robotic arm with a gripper as the end-effector. The robot movement speed is limited to a slow 100 millimeters per second. Thus, the execution time is slow in comparison.

5.4 Number of Execution Errors

Execution errors can be any mistakes that occur during the execution of the remote robot. Misplacement of the rectangular-prism or any of the nine finger joints are the possible execution errors that the three-fingered hand IBTS can have. The misplacement of the rectangular-prism is more likely because currently the physical fingertips' contact to the rectangular-prism is not reinforced by any means. The fingers sometimes cannot grip onto the rectangular-prism and even when the fingers are moved to the location as specified, it may not lead to the rectangular-prism to be moved accordingly. Instead, the rectangular-prism remains at its original spot or gets tilted to either side or tilted back and forth. The finger joints themselves are mostly executed correctly but note that there are approximately 150 different positions that the servomotors can reach and from -45° to +45 ° range of each finger joints and that there are infinite non-integer joint angles that each finger joint can reach. Any floating-point number between –45 to +45, have to be mapped to the 150 different positions of the servomotor. +45 ° is mapped to position 0, 0 ° is mapped to position 75 and –45 ° is mapped to position 150. At +22.5 °, it cannot be exactly half way between +45 ° and 0 °, at position 37.5. This is called quantization error. There is no physical location 37.5 that the servomotor can reach, it has to be mapped to position 38. Likewise, all other non-integer representations of the joint angles have to be rounded to the nearest position. Although this three-fingered hand's control input can specify precise input, it cannot be physically accomplished due to the physical limits of the servos. Thus, execution errors of the fingers are created. It is to note that this quantization error is small in comparison to slipping and inflexibility of the fingers. Execution errors are not measured in Talyor's IBTS, it is hard to do because these errors

rely on the users' feedback. If users do not see or report the error, then none can be documented because the system cannot automatically detect them. Furthermore, sometimes these errors may occur but the users may think that those are human control errors and not errors of the system itself.

5.5 Reliability – 24 hours a day, 7 days a week

Each IBTS must be reliable such that it can be available 24 hours a day and 7 days a week while requiring minimal maintenance. This is because Internet users around the world are located in different time-zones. Internet users may access each IBTS at any time of the day. Also, systems that require a lot of down-times will soon be abandoned by frequent users; users do not like to return to sites that are unpredictable and unreliable. The three-fingered hand system is reliable; it has been up since July 1 and it has not required any maintenance yet. During the month of July, the hand has been controlled more than 500 times. In the future, problems may occur with the servomotors due to wearing of the armature; the only moving parts of the motors. Although the three-fingered hand is fresh and cannot be compared fairly to the Australian system in terms of reliability, it is interesting to note that the Australian IBTS has been up since 1995. At the beginning, the Australian system suffered an average of 50% downtime. Since then, the system has made a lot of improvements. The system now only has 5% downtime. It happens when the robot operating system does not recognize its hard disk or any of its programs for no particular reason. As time passes, it will better reflect the true reliability of the three-fingered hand system and more accurate comparisons can be made with the Australian system.

5.6 Subjective Evaluation Criteria

The above evaluation criteria for IBTS can be measured in terms of numbers and can be analyzed easily. However, there are features of an IBTS that cannot be quantitatively measured. For example, one cannot easily quantify the "feel" of telepresence that an IBTS gives to a user or measure the ease of use of an application-specific control interface.

It is important that a user has a strong feel of telepresence when controlling a remote robot because that way, the user can quickly react to any environmental changes in the remote location. The remote location is not static once there is a robot in the scene that manipulates the environment. Even if the robot does not manipulates and simply observes, unexpected changes may take place in the remote location and will require the human operator to react accordingly. If the user feels as if he or she is present in that remote location, reaction time increases. In this IBTS, the user only observes what the live-camera shows and because of the video delay, the user definitely does not have a great scene of telepresence of the remote location.

Another question is: *How does one measures the ease of use of an interface?* This is possible if an IBTS has more than one control interface to compare with. For example, the Australian robot has both a text-based input and a graphical input interface that perform the exact same set of functions. Then a user survey can be used to analyze their preferences. However, like the previous criteria, these are subjective measures. Different user feels differently based on ones' background and familiarity with the system, with Internet or with computers in general. To quantify these qualities of an IBTS is yet to be standardized.

Chapter 6

Conclusion

6.1 Summary

In this research, we have defined a system architecture for Internet-based teleoperation system. This architecture is essential to the design and development of any future IBTS. To show the feasibility of this architecture, we presented the first Internet-based teleoperation system that allows users worldwide to control a three-fingered hand located in the Robotics Laboratory of the University of Alberta. This three-fingered hand can be manipulated to move freely or to hold onto a box and move it in a three-dimensional space. In order to construct such a system, we also examined the ever-changing Internet technologies: Java and VRML. We further pointed out the advantages and disadvantages of using these technologies. The use of VRML is unique and it provides a new input format for users to remotely control the hand. This VRML three-dimensional model of the remote environment is one of its kind because it serves as both input and output. It allows users to input the next move of the hand through the VRML model. It also shows the users the status of remote environment. The model reflects the up-to-date location of all the objects that are in the scene of the environment. This research is robotic-based but it also examines the different graphical theories likes plane equations, back face

elimination and z-buffering. These computer graphics techniques are applicable to Internet-based teleoperation system because they can be used to find an optimal viewing direction for the users. Finally, evaluation criteria for IBTS are presented. The criteria were also applied on the three-fingered hand system. It is shown that the system is competitive but there is room for improvement in system speed and quality of the video display.

6.2 Future Research

In this research, a number of issues in Internet-based teleoperation system have been examined. However, there are several areas that can be looked at more closely. First, it would be interesting to create another IBTS using the defined architecture and test if that can be accomplished by only modifying parts of the three-fingered hand system. Success would show that the architecture is applicable to any typical IBTS and the existing system is modular and flexible to changes.

Also, finding an optimal view-angle can be expanded to include the automatic relocation of the camera. It is best if the camera will continuously move to an optimal view-angle and the live-video display can then show an optimal picture at all times as the hand and the box are being manipulated.

It will also be beneficial to develop an ideal programming vehicle for developing IBTS. Currently, Internet technologies change rapidly, but they are not ideal for teleoperation system because for any Internet-based IBTS, it is best if the programming vehicle is platform-independent, object-oriented, capable of 3D modeling, has network capability, speedy, easy to use and end-user friendly. If there is such a programming

language, it will be easy to implement various Internet-based teleoperation system. This language appears to be an improved version of Java that has all the features of VRML and has been sped up. It is important that the programming vehicle is platform-independent because for Internet-based applications, users are diverse and one cannot confine a user to have a specific system in order to use the applications. This will limit the audience and usability of the applications. Object-oriented features are also beneficial. In today's applications, a lot of system components are required and components need to be flexible such that creation and modification to the components can be done with minimal effort and time. Three-dimensional modeling capability is also necessary because applications are becoming more complicated everyday and three-dimensional tele-operational tasks are inevitable and thus, it requires a 3D representation of the remote robot and environment. Such ideal programming vehicles also need to be able to establish connections between the remote users and remote environment; connections are also required between the remote host machine and the robot itself. The ability to network is a must for an ideal Internet-based programming language. Because of the distance between the users and the remote robots, there may be time delay in the Internet transfers. In order to minimize any time delay, the programming vehicle itself has to be speedy and must not add to the delay in Internet traffic. Currently, that is the major flaw in the programming language – Java. This ideal vehicle also needs to be user-friendly, both to the programmers and the users. For the programmers, the programming vehicle should be easy to learn and to create applications with ease. For the users, the resulted applications that are created using this ideal programming language should be pleasant to use and provides great controls for the users. Overall, these criteria seem to

describe an improved version of Java and with the ever-changing release versions in Java, all of the criteria may be met in the near future.

In this thesis, we defined the architecture of an Internet-based teleoperation system and proven that it is sound. This also marks the first multi-fingered hand system on the Internet. Furthermore, this robotic research introduced the use of graphics theories in finding an optimal viewing-angle for an IBTS. Although many IBTSs are for recreational use, the real interest in studying IBTS is to one day allow humans to be virtually present in any location of the universe.

Bibliography

[Anand] Vera B. Anand, "Computer Graphics and Geometric Modeling for Engineers", 1993, John Wiley & Sons, Inc.

[Atkinson] D. Atkinson, P. Ciufo, S. Krav, "Robotoy – Technical Details", 1998, http://robotoy.elec.uow.edu.au/tech.html

[Backes, 1997] P. Backes, G. Tharp, K. Tso, "The Web Interface for Telescience (WITS)", IEEE International Conference on Robotics & Automation 1997

[Backes, 1998] P. Backes, K. Tso, G. Tharp, "Mars Pathfinder Mission Internet-Based Operations Using WITS", IEEE International Conference on Robotics & Automation 1998

[Carnegie] Carnegie Science Center, "New Controller Page for Telerobot", 1998, http://csc.clpgh.org/telerobt/ex1.htm

[Carter] M. Carter, "Using Telerobotics for Recreation", International Conference on Field and Service Robotics, 1997

[Connectix] http://www.connectix.com

[Craig] John J. Craig, "Introduction to Robotics – Mechanics and Control", 1986, Addison-Wesley

[DePasquale] P. DePasquale, J. Lewis, M. Stein, "A Java interface for asserting interactive telerobotic control", IEEE/RSJ International Conference on Intelligent Robotic Systems 1998, http://yugo.mme.wilkes.edu/~villanov

[Digital] Digital Media Arts, "The Eyebot Project", 1996, http://www.dma.nl/eyebot/

[Drascic] D. Drascic, "ETC-Lab: Telerobotics Research", 1996, http://vered.rose.toronto.edu/projects/telerobotics.html

[Fiorini] P. Fiorini, R. Oboe, "Internet-Based Telerobotics: Problems and Approaches", ICAR 1997, http://robotics.jpl.nasa.gov/people/fiorini/work.html

[Fisher] Fisher Jason Team, "Jason@Fisher", 1998, http://149.69.43.7/jason/

[Goldberg, K. 1994] K. Goldberg, M. Mascha, "Beyond the Web: Excavating the Real World Via Mosaic", 1994, http://www.usc.edu/dept/raiders/paper/

[Goldberg, K. 1995] K. Goldberg, M. Mascha, "Desktop Teleoperation via the World Wide Web", ICRA 1995.

[Goldberg, K. 1997] K. Goldberg, J. Santarromana, "About the Tele-Garden", 1997, http://telegarden.aec.at/html/intro.html

[Goldberg, S.] S. Goldberg, G. Bekey, M. McLaughlin, "The javamuse Applet: USC Interactive Art Museum", http://digimuse.usc.edu/robot/

[Harrington] Steven Harrington, "Computer Graphics – A Programming Approach", 1987, McGraw-Hill

[Hill] Francis S. Hill, Jr. ,"Computer Graphics", 1990, Macmillan

[Johnsen] Edwin G. Johnsen and William R. Corliss, s"Human Factors Applications in Teleoperator Design and Operation", 1971, Wiley

[Kawabata] K. Kawabata, T. Ishikawa, T. Fujii, T. Noguchi, H. Asama, I. Endo, "Teleoperation of Autonomous Mobile Robot under Limited Feedback Information", International Conference on Field and Service Robotics, 1997

[Kehoe] C. Kehoe, J. Pitkow and J. Rogers, "GVU's Ninth WWW User Survey Report", July 1998, http://www.cc.gatech.edu/gvu/user_surveys/survey-1998-04/

[Hand] C. Hand, "A Survey of 3D Interaction Techniques", Department of Computing Science, De Montfort University, The Gateway, Leicester LE1 9BH, UK, cph@dmu.ac.uk

[Hanson] A. Hanson, E. Wernert, "Constrained 3D Navigation with 2D Controllers", IEEE Proceeding of Visualization 1997

[Kenney] P. Kenney, T. Satio, "Results of a Survey on the Use of Virtual Environment Technology", 1994, http://www.vetl.uh.edu/Hubble/longpaper.html

[Marrin, 1996] C. Marrin, "Anatomy of a VRML Browser", 1996, http://www.marrin.com/vrml/Interface.html

[Marrin, 1997] C. Marrin, "Proposal for a VRML 2.0 Informative Annex", 1997, http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html

[Moss] K. Moss, Java Servlets, McGraw-Hill, 1998

[NASA] NASA, "Selected Robotics Resources on the Internet", http://ranier.oact.hq.nasa.gov/telerobotics_page/internetrobots.html

[Nehmzow] U. Nehmzow, A. Buhlmeier, H. Durer, M. Nolte, "Remote Control of Mobile Robot via Internet", 1996, Department of Computing Science, University of Manchester, Technical Report Series UMCS-96-2-3

[Paulos] E. Paulos, J. Canny, "Delivering Real Reality to the World Wide Web via Telerobotics", ICRA 1996

[Pesce] M. Pesce, "VRML and Java: A Marriage Made in Heaven", 1998, http://developer.netscape.com/viewsource/pesce_vrml2/pesce_vrml2.html

[Popular] Popular Mechanics, "Robot Hand Signals", 1995, http://popularmechanics.com/popmech/sci/tech/9504TURBWM.html

[Richardson] Doug Richardson, "Unmanned Aerial Vehicle (UAV) and battlefield robots ", http://www.atalink.co.uk/DSR/CLIENT/richards.htm

[Saucy] P. Saucy, F. Mondada, "KhepOnTheWeb", an Example of Remote Access to Mobile Robotics Facilities, IROS 1997, http://lamipc20.epfl.ch/Docs/Telemanipulation/index.htm

[SSC] SSC San Diego Robotics, "Submersible Cable-Actuated Teleoperator", http://www.spawar.navy.mil/robots/undersea/scat/scat.html

[Sherdian] T. B. Sheridan, Telerobotics, Automation and Human Supervisory Control, MIT press, 1992

[SNWSC] Space and Naval Warfare Systems Center, San Diego, "Airborne Remotely Operated Device (1982-1988)", http://www.spawar.navy.mil/robots/air/arod/arod.html

[Su] S. Su, R. Furuta, "VRML-based Representations of ASL Fingerspelling on the World-Wide Web", http://www.csdl.tamu.edu/~su/publications/assets98.html

[Sun] Sun Microsystems, Inc., "Java 3D FAQ", http://java.sun.com/products/java-media/3D/forDevelopers/java3dfaq.html

[Sun, 1998] Sun Microsystems, Inc., "Java Technology Home Page", 1998, http://java.sun.com

[Taylor, 1995] K. Talyor, J. Trevelyan, "A Telerobot on The World Wide Web", National Conference of the Australian Robot Association 1995, http://telerobot.mech.uwa.edu.au/robot/telerobo.htm

[Taylor, 1997] K. Taylor, "Issues in Internet Telerobotics", International Conference on Field and Service Robotics 1997, http://telerobot.mech.uwa.edu.au/ROBOT/anupaper.htm

[Wolf] H. Wolf, "From Web Server to Railroad Layout", http://rr-vs.informatik.uni-ulm.de/rr/LayoutControl.html

[QuickCam] QuickCam VC, http://www.logitech.com/Cameras/quickcamvc_moreinfo.html